AFIT/GA/ENG/92J-01

MULTIPLE MODEL ADAPTIVE ESTIMATION APPLIED TO THE VISTA F-16 WITH ACTUATOR AND SENSOR FAILURES
VOLUME II

THESIS

Timothy E. Menke

AFIT/GA/ENG/92J-01

AFIT/GA/ENG/92J-01

MULTIPLE MODEL ADAPTIVE ESTIMATION APPLIED TO THE VISTA F-16 WITH ACTUATOR AND SENSOR FAILURES

VOLUME II

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fullfillment of the

Requirements for the Degree of

Master of Science in Aerospace Engineering

Timothy E. Menke, B.S.A.E.

June 1992

Approved for public release; distribution unlimited

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	1 Jun 92	Final

4. TITLE AND SUBTITLE

Multiple Model Adaptive Estimation Applied to the VISTA F-16 with sensor and actuator failures, Volume II - Appendices

5. FUNDING NUMBERS

6. AUTHOR(S)

Timothy E. Menke

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Wright-Patterson AFB, Ohio

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GA/ENG/92J-01

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

WL/FIGL
Wright-Patterson AFB, Ohio

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unlimited distribution

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

A Multiple Model Adaptive Estimation (MMAE) algorithm is applied to the Variable Stability In-flight Simulator Test Aircraft (VISTA) F-16 at a low dynamic pressure flight condition (0.4 Mach at 20000 ft). A complete F-16 flight control system is modeled containing the longitudinal and lateral-directional axes. Single and dual actuator and sensor failures are simulated including: complete actuator failures, partial actuator failures, complete sensor failures, increased sensor noise, sensor biases, dual complete actuator failures, dual complete sensor failures, and combinations of actuator and sensor failures. Failure scenarios are examined in both maneuvering and straight and level flight conditions. Single scalar residual monitoring techniques are evaluated with suggestions for improved performance. A hierarchical "moving bank" structure is utilized for multiple failure scenarios. Simultaneous dual failures are included within the study. White Gaussian noise is included to simulate the effects of atmospheric disturbances, and white Gaussian noise is added to the measurements to simulate the effects of sensor noise.

14. SUBJECT TERMS

Kalman Filtering, Multiple Model Adaptive Estimation, MMAE, Failure detection and isolation, Bayesian, MMAC, Multiple Model Adaptive Control

15. NUMBER OF PAGES 244

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev 2-89)
Prescribed by ANSI Std 239-18

Acknowledgements

Throughout my studies I have had the pleasure of working with many outstanding instructors, but these experiences pale in comparison to the lessons learned by working with my thesis advisor, Dr. Peter S. Maybeck. Dr. Maybeck provides a challenging, dynamic, and inspiring learning environment. I firmly believe I became an engineer in this thesis effort, under his guidance. I will never forget his optimism, kindness, and patience. It has been an honor to work with him. Also, I wish to thank the other two members of my thesis committee, Major Riggins and Captain Ridgely. Their comments and questions significantly improved the quality of the final report. I want to thank my sponsor, Captain Stuart Sheldon and WL/FIGL. Also, I'd like to thank Capt Danny "the computer wizard" Shoop for his help in those rough coding spots early on in the development (I didn't forget you). Finally, I want to thank my "home" crew for holding down the fort while I was "goofing off" (working). They are responsible for maintaining my fighting spirit. Thank you Heidi and Gunnar.

Timothy E Menke

TABLE OF CONTENTS

List of Figures

Abstract

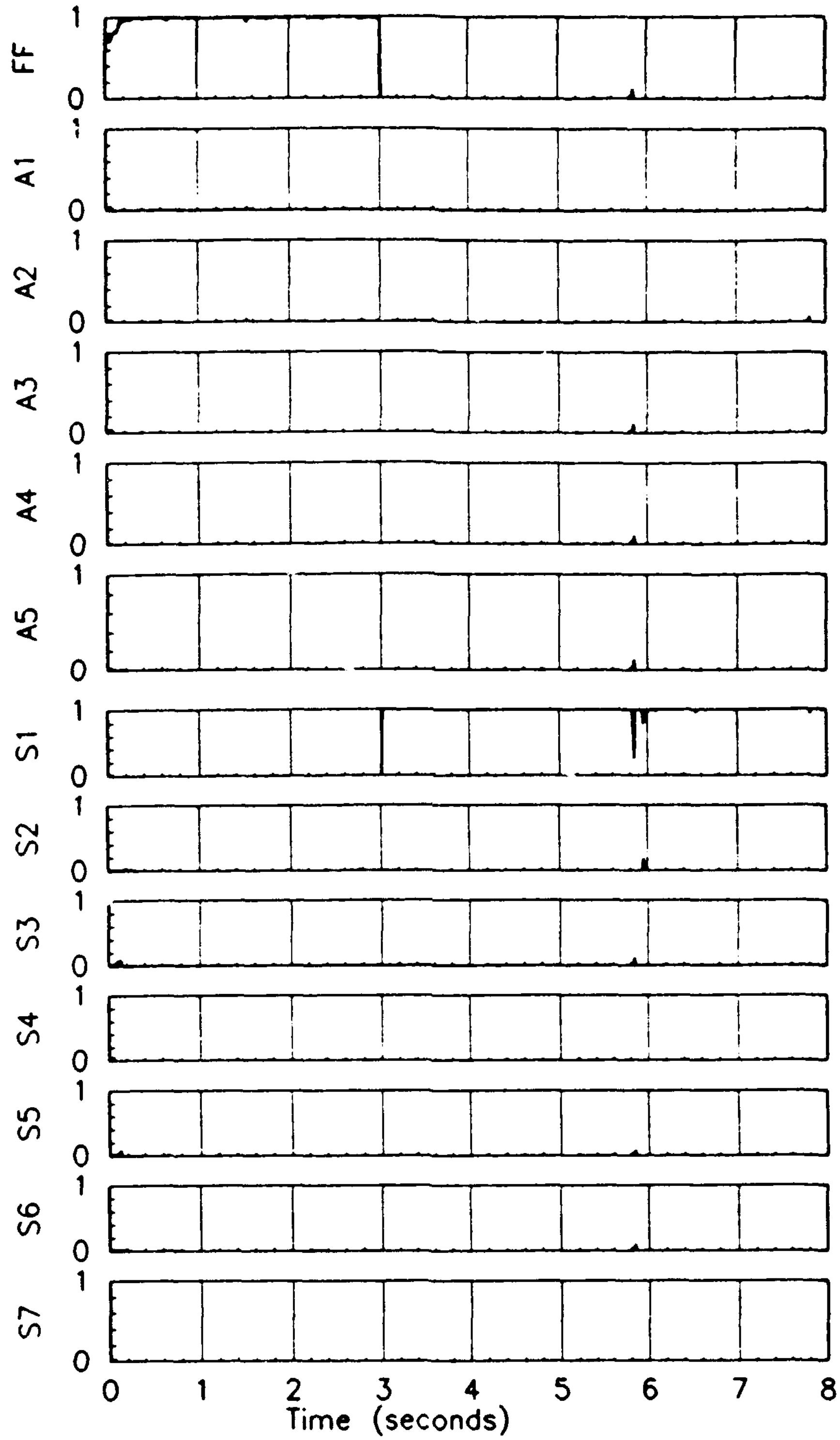
A Multiple Model Adaptive Estimation (MMAE) algorithm is applied to the Variable Stability In-flight Simulator Test Aircraft (VISTA) F-16 at a low dynamic pressure flight condition (0.4 Mach at 20000 ft). A complete F-16 flight control system is modeled containing the longitudinal and lateral-directional axes. Single and dual actuator and sensor failures are simulated including: complete actuator failures, partial actuator failures, complete sensor failures, increased sensor noise, sensor biases, dual complete actuator failures, dual complete sensor failures, and combinations of actuator and sensor failures. Failure scenarios are examined in both maneuvering and straight and level flight conditions. The system performance is characterized when excited by purposeful commands and dither signals. Single scalar residual monitoring techniques are evaluated with suggestions for improved performance. A Kalman filter is designed for each hypothesized failure condition. In this thesis, thirteen elemental Kalman filters are designed encompassing: a no failure filter, left stabilator failure filter, a right stabilator failure filter, a left flaperon failure filter, a right flaperon failure filter, a rudder failure filter, a velocity sensor failure filter, an angle of attack sensor failure filter, a pitch rate sensor failure filter, a normal acceleration sensor failure filter, a roll rate sensor failure filter, a yaw rate sensor failure filter, and a lateral acceleration sensor failure filter. The Bayesian Multiple Model Adaptive Estimator (MMAE) algorithm blends the state estimates from each of the filters, representing a hypothesized failure, multiplied by the filters computed probability. The blended state estimates are sent to the VISTA F-16 flight control system. A hierarchical "moving bank" structure is utilized for multiple failure scenarios. Simultaneous dual failures are included within the study. White Gaussian noise is included to simulate the effects of atmospheric disturbances, and white Gaussian noise is added to the measurements to simulate the effects of sensor noise. Each elemental Kalman filter is compared to the truth model with a selected failure. Filters with residuals that have mean square values most in consonance with their internally computed covariance are assigned the higher probabilities.

APPENDIX A: ADDITIONAL RESULTS FOR FAILURES

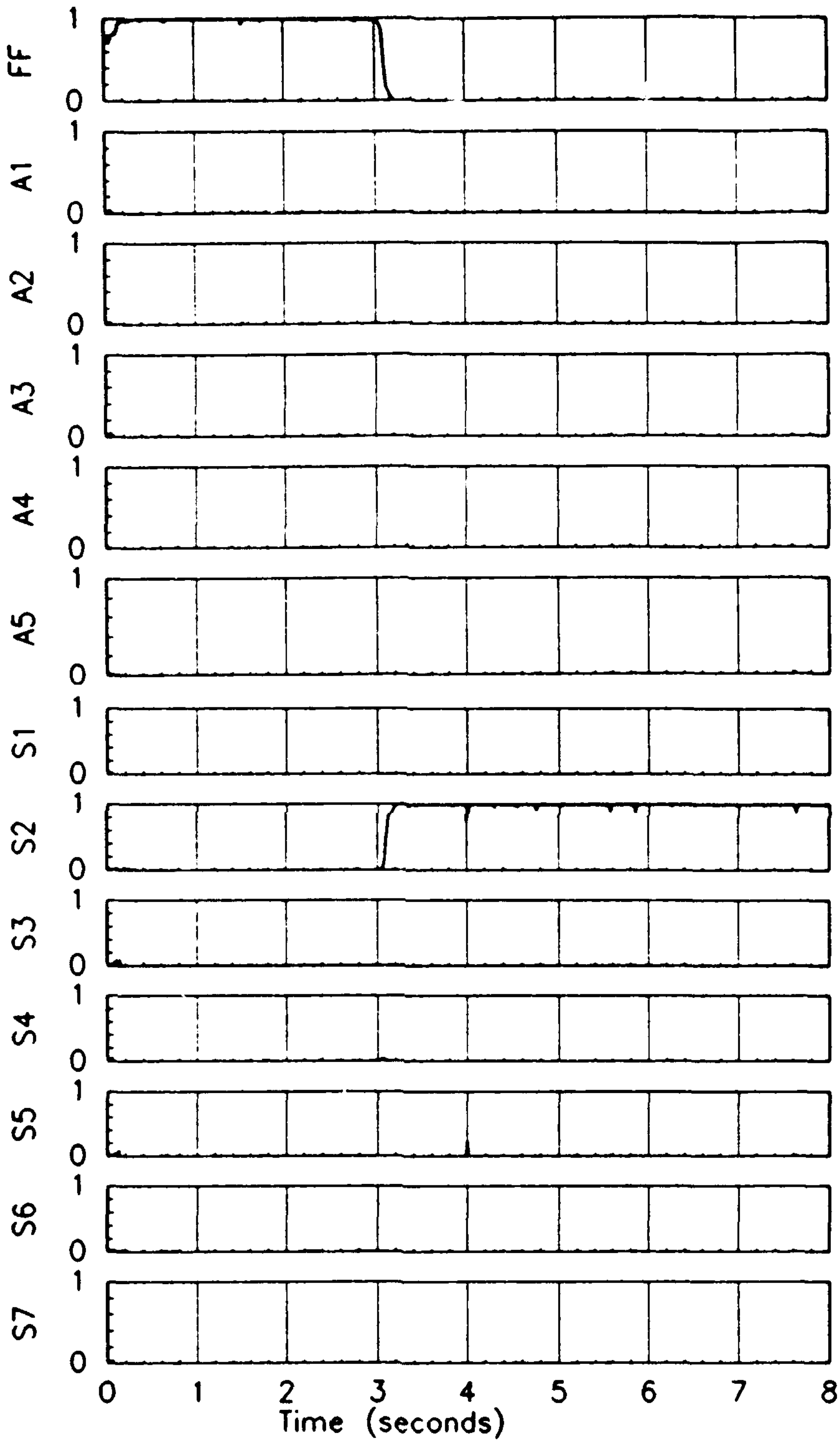
This appendix contains the remainder of the multiple failure data not presented within Chapter 4. The data is organized similar to Chapter 4. All of the data presented within this appendix is for hard actuator and sensor failure combinations. Figures A.1 through A.6 present the remainder of the data, not presented in Chapter 4, for the second subliminal dither signal. Figures A.7 through A.13 present data for a sinusoidal dither signal. Figures A.14 through A.20 present data for a purposeful roll command. Figures A.21 through A.29 present data for a purposeful roll and pull command. Figures A.30 through A.43 present data for a purposeful rudder kick and hold. Preceding each of set of figures is a two page Fortran description of the command input. Note sections of the code are commented out depending on the signal implemented. This allows the reader to evaluate the command signal's relative timing and magnitude.

```
      SUBROUTINE COMMAND(FBC,FAC,FPC,T)
c ----- Provides the vista flight control system [CNTRL.FOR]
c       with the command signals for the longitudinal axis
c       [FBC], the lateral command [FAC], and the directional
c       command [FPC].  Additionally, simulation results have
c       indicated the need for a dither signal to "shake up"
c       the system and aid the filters in their identification
c       tasks.

      INCLUDE 'DECLARR.TXT'
      REAL FBC,FAC,FPC,DON,T,omega1,omega2,omega3


      SAVE

c     COMMAND SIGNALS


c.......................
c  Dither signal generation
c    --NOTE:  if a control command is to be used,
c             it must be added to the below
c             created dither signal.
c.......................
c
c
c   P U L S E D   D I T H E R   S I G N A L
c
c             NON - SUBLIMINAL
c.......................

       Don=amod(t,3.0)

       IF((Don.ge.0.0).and.(Don.lt.0.125))THEN
         FBC = 13.5
         FAC = -7.5
         FPC = 24.0
        ELSE IF((Don.ge.0.125).and.(Don.lt.0.25))THEN
          FBC =-13.0
          FAC = 7.5
          FPC =-24.0
        ELSE
          FBC = 0.0
          FAC = 0.0
          FPC = 0.0
        END IF
c
c.......................
c
c
c   P U L S E D   D I T H E R   S I G N A L
c
c             SUBLIMINAL
c.......................
c

c       Don=amod(t,3.0)

c       IF((Don.ge.0.0).and.(Don.lt.0.125))THEN
c          FBC = 15.2
c          FAC = -7.0
c          FPC = 22.0
c        ELSE IF((Don.ge.0.125).and.(Don.lt.0.25))THEN
c          FBC =-16.5
c          FAC =  8.0
c          FPC = 22.0
c        ELSE
c          FBC = 0.0
c          FAC = 0.0
c          FPC = 0.0
c        END IF

c
c..............................
c
c   S I N U S O I D A L   D I T H E R   S I G N A L
c
c..............................
c

c   FREQUENCY

        omega1 = 15.0
        omega2 = 15.0
        omega3 = 15.0
```

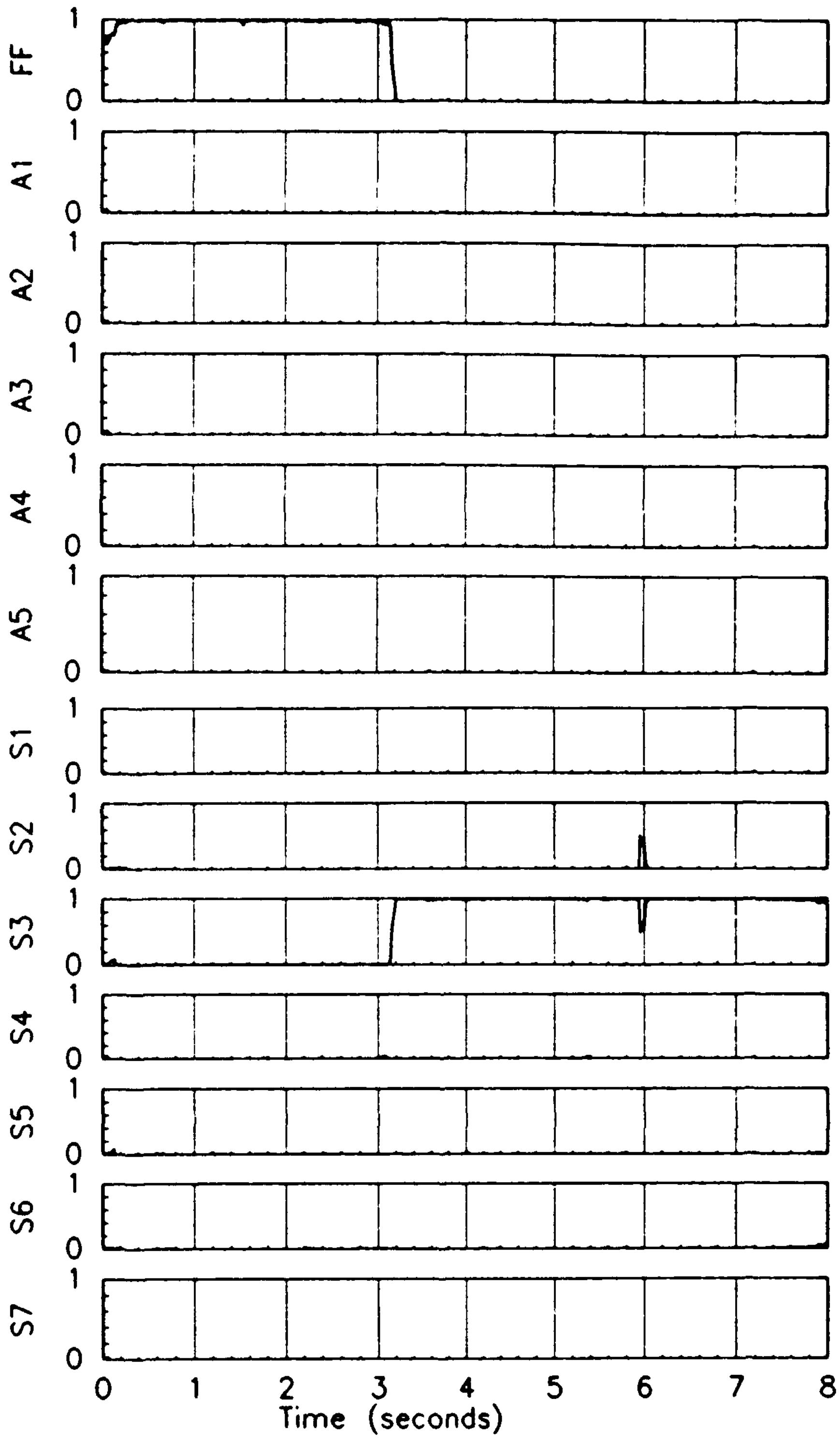
```
c     SIGNAL

c           FBC= 12.0*sin(omega1*t)
c           IF(FBC.LT.0.0)FBC=12.5*sin(omega1*t)
c           FAC= -11.0*sin(omega2*t)
c           FPC= 30.0*sin(omega3*t)

c  . . . . . . . . . . . . . . . . . . . . . . . . . . .
c
c
c  U S E R   D E F I N E D   D I T H E R   S I G N A L
c
c
c  . . . . . . . . . . . . . . . . . . . . . . . . . . .

c      Don=amod(t,3.0)

c      IF((Don.ge.0.0).and.(Don.lt.0.1))THEN
c        FBC = 15.2
c        FAC = 16.0
c        FPC = 55.0
c      ELSE IF((Don.ge.0.1).and.(Don.lt.0.125))THEN
c        FBC = (-15.2/0.025)*(Don - 0.1) + 15.2
c        FAC = (-16.0/0.025)*(Don - 0.1) + 16.0
c        FPC = (-55.0/0.025)*(Don - 0.1) + 55.0
c      ELSE IF((Don.ge.0.125).and.(Don.lt.0.3))THEN
c        FBC =-16.5
c        FAC =-14.0
c        FPC =-50.0
c      ELSE IF((Don.ge.0.3).and.(Don.lt.0.325))THEN
c        FBC = (16.5/0.025)*(Don - 0.3) - 16.5
c        FAC = (14.0/0.025)*(Don - 0.3) - 14.0
c        FPC = (50.0/0.025)*(Don - 0.3) - 50.0
c      ELSE
c        FBC = 0.0
c        FAC = 0.0
c        FPC = 0.0
c      END IF

c  . . . . . . . . . . . . . . . . . . . . . . . . . . .
c
c
c P I L O T   P U R P O S E F U L   C O M M A N D S
c
c
c  . . . . . . . . . . . . . . . . . . . . . . . . . . .

c     Pitch Path

c       IF((t.GT.2.95).and.(t.LT.3.45))THEN
c        FAC=5.0
c        FAC=20.0
c        FPC=-40.0
c       ENDIF

c     Roll Path


c     Yaw Path

      RETURN
      END
```



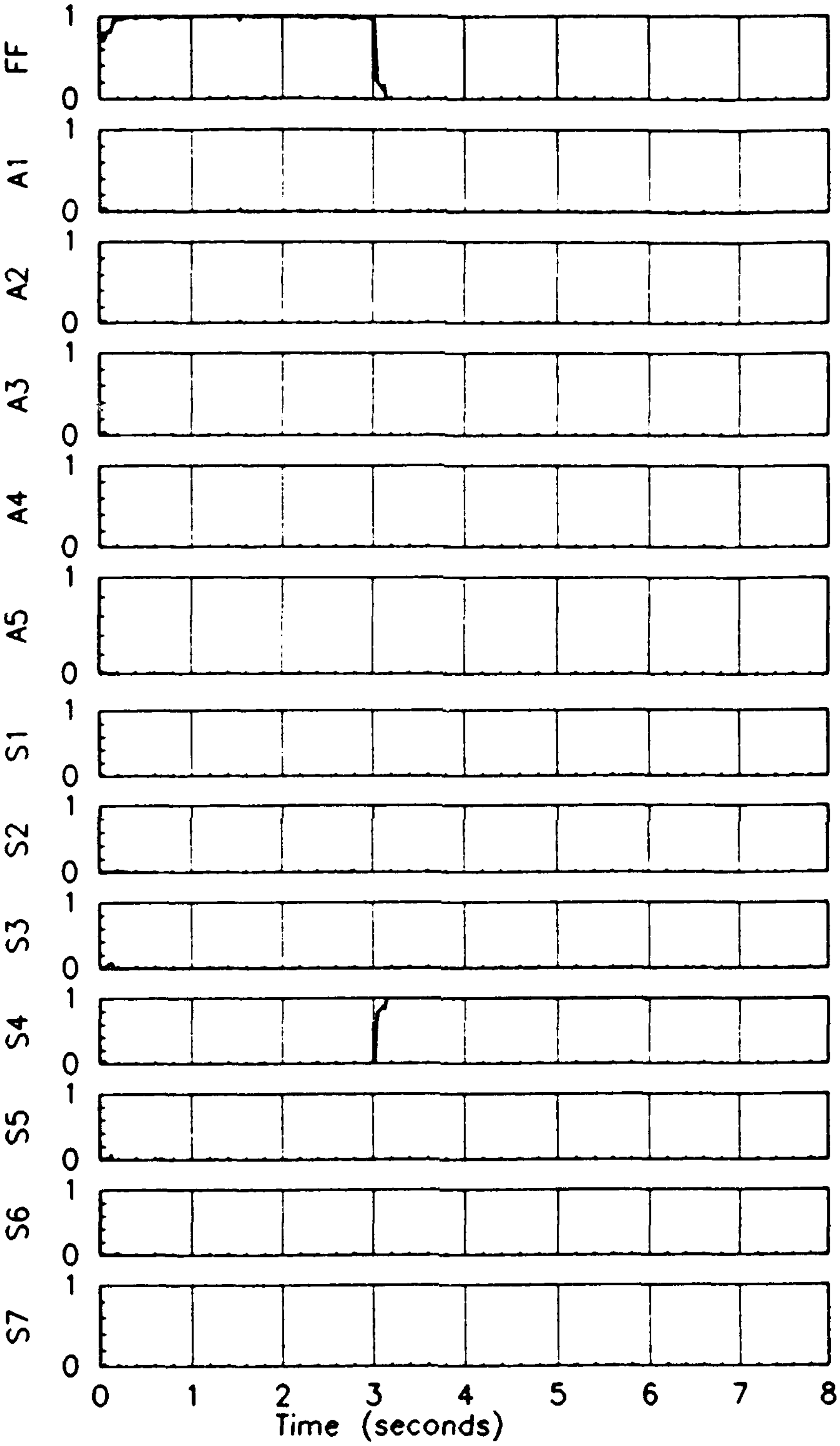
A.1 Probabilities for a velocity sensor failure using subliminal dither signal #2



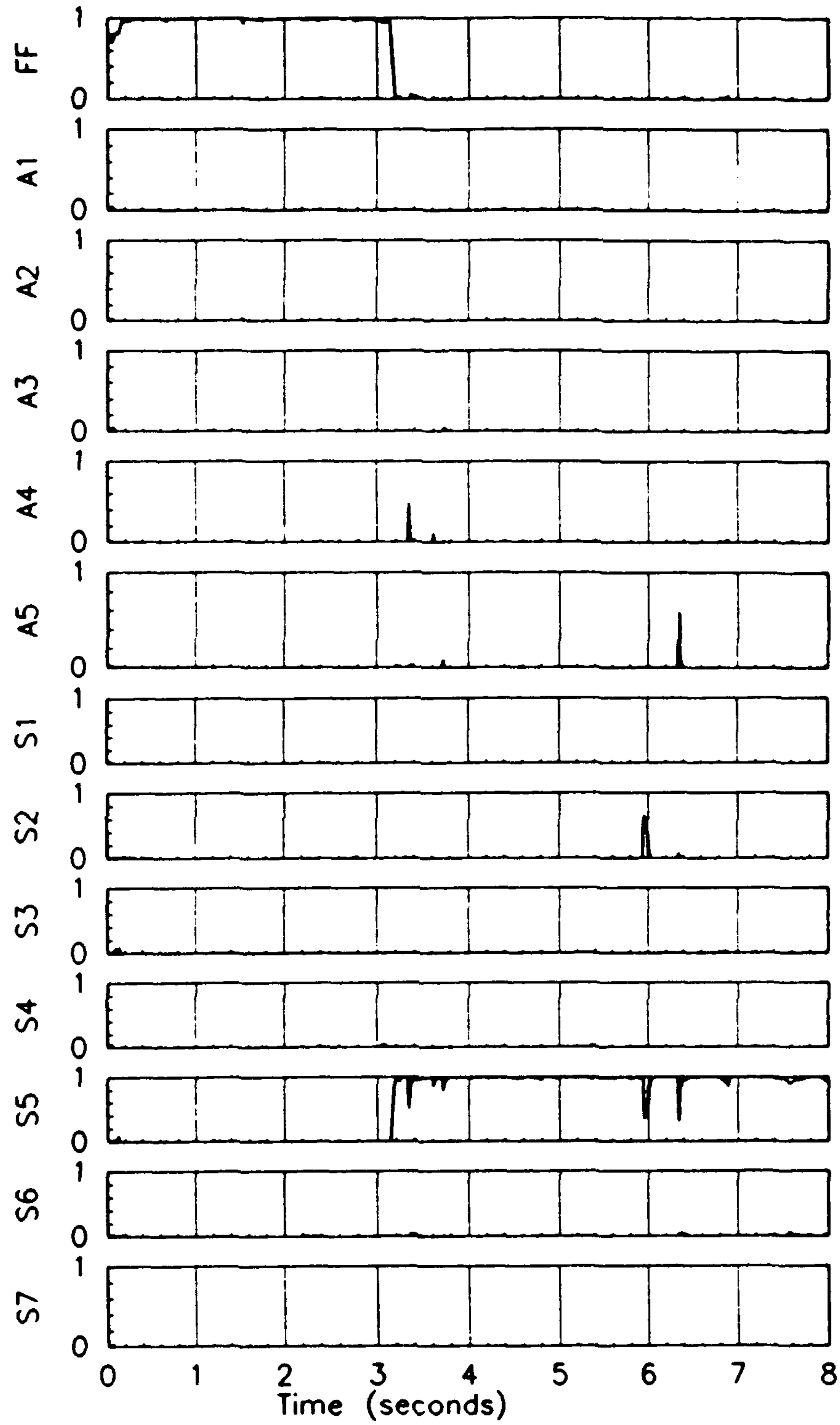
A.2 Probabilities for a angle of attack sensor failure using subliminal dither signal #2



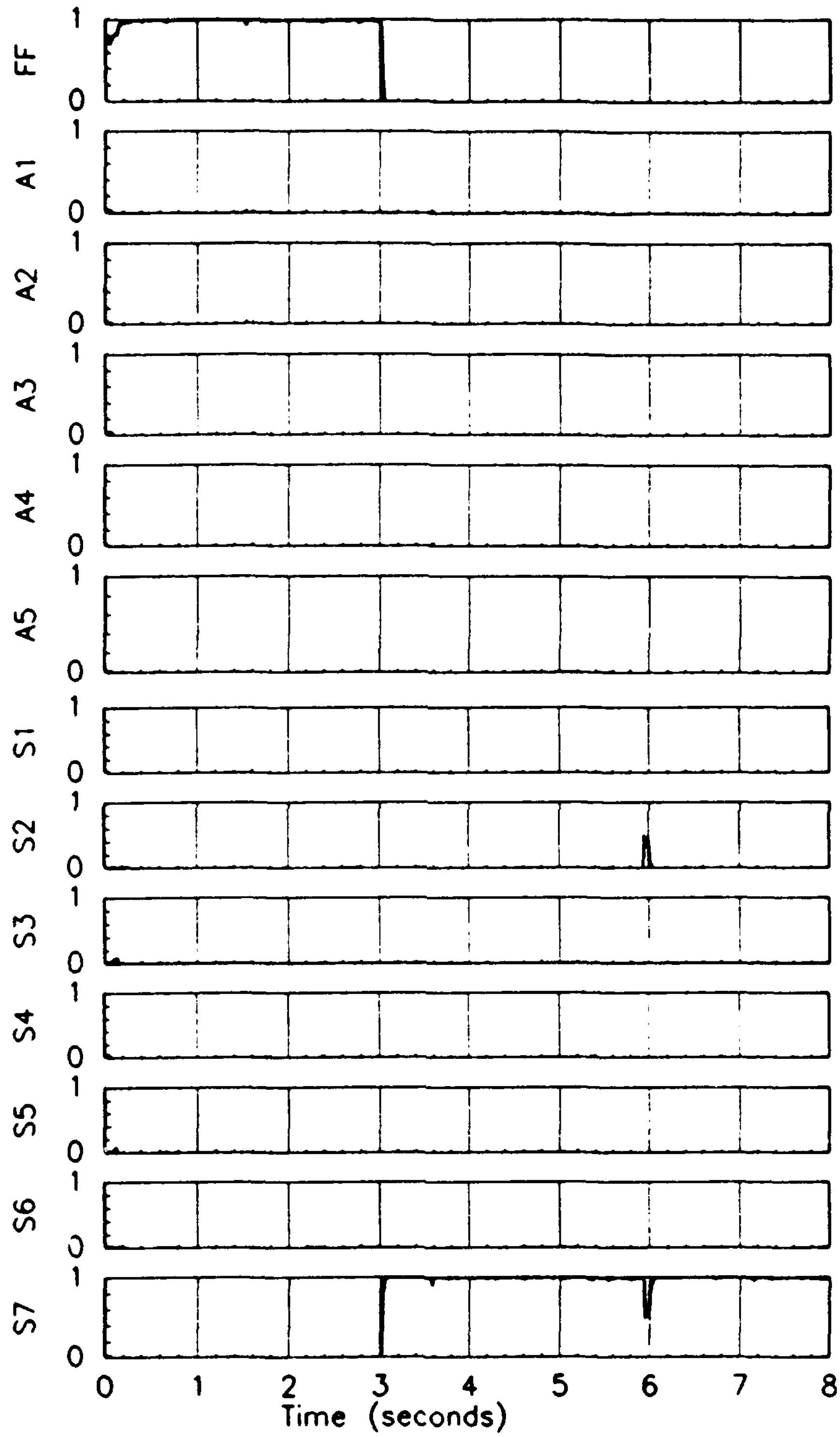
A.3 Probabilities for a pitch rate sensor failure using subliminal dither signal #2



A.4 Probabilities for a normal acceleration sensor failure using subliminal dither signal #2



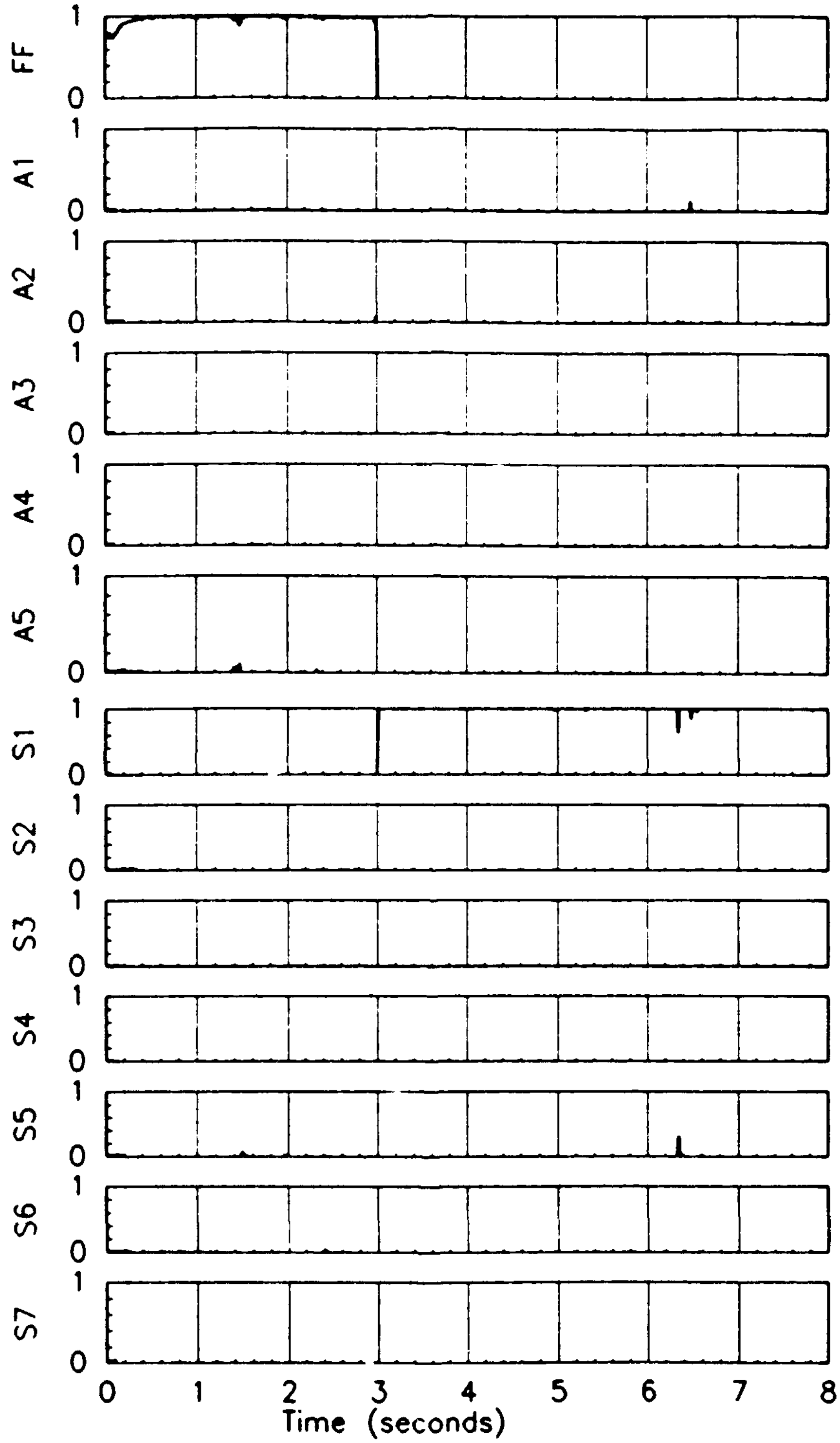
A.5 Probabilities for a roll rate sensor failure using subliminal dither signal #2



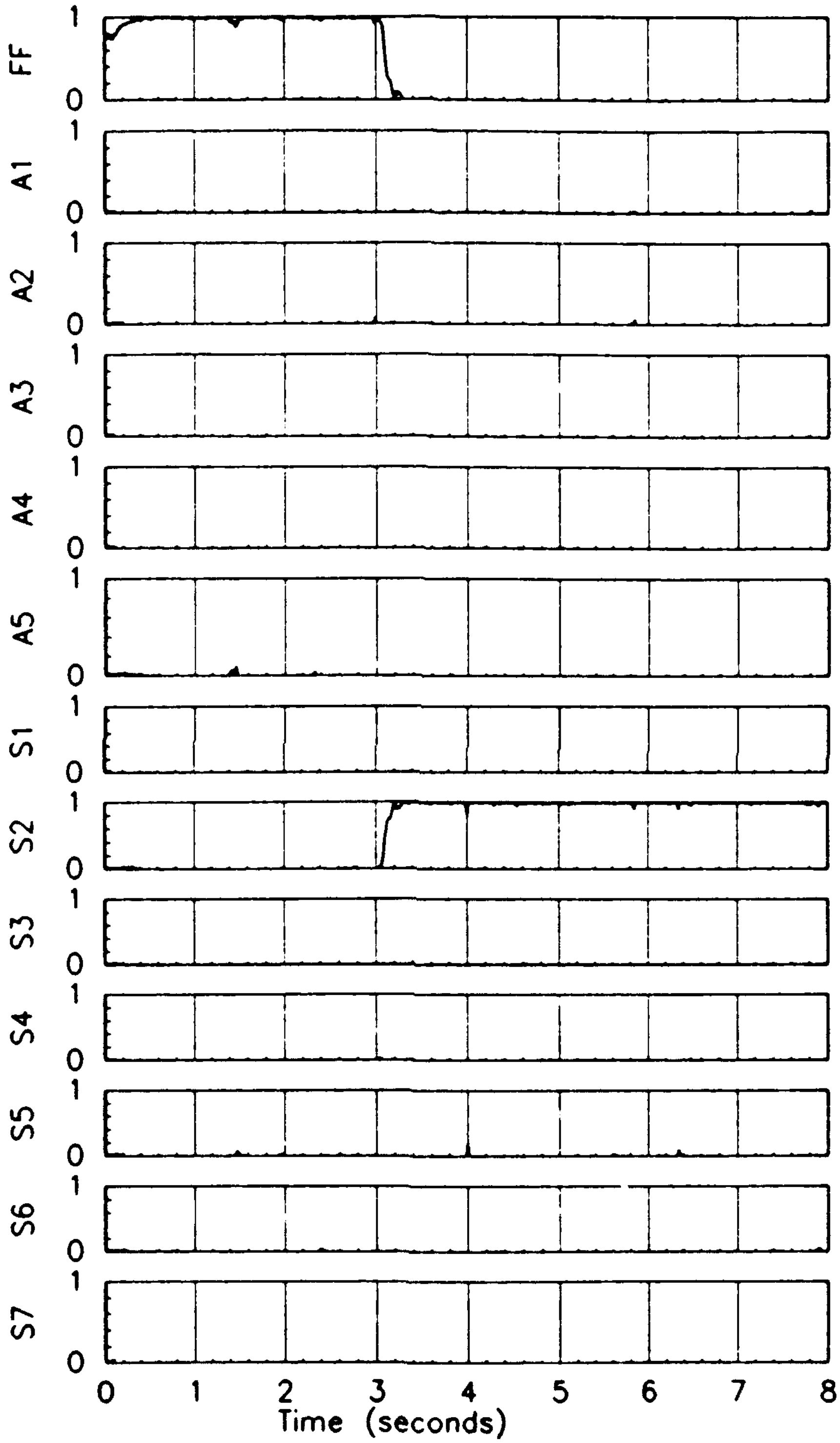
A.6 Probabilities for a lateral acceleration sensor failure using subliminal dither signal #2

```
      SUBROUTINE COMMAND(FEC,FAC,FPC,T)
c ---- Provides the vista flight control system (CNTRL.FOR)
c      with the command signals for the longitudinal axis
c      (FEC), the lateral command (FAC), and the directional
c      command (FPC).  Additionally, simulation results have
c      indicated the need for a dither signal to "shake up"
c      the system and aid the filters in their identification
c      tasks.

      INCLUDE 'DECLARR.TXT'
      REAL FEC,FAC,FPC,DON,T,omega1,omega2,omega3


      SAVE

c     COMMAND SIGNALS


c. . . . . . . . . . . . . . . . . . . . . . .
c  Dither signal generation
c     --NOTE:  if a control command is to be used,
c              it must be added to the below
c              created dither signal.
c. . . . . . . . . . . . . . . . . . . . . . .
c
c
c    P U L S E D   D I T H E R   S I G N A L
c
c              NON - SUBLIMINAL
c. . . . . . . . . . . . . . . . . . . . . . .

c       Don=amod(t,3.0)

c       IF((Don.ge.0.0).and.(Don.lt.0.125))THEN
c         FEC = 13.5
c         FAC = -7.5
c         FPC = 24.0
c        ELSE IF((Don.ge.0.125).and.(Don.lt.0.25))THEN
c         FEC =-13.0
c         FAC = 7.5
c         FPC =-24.0
c        ELSE
c         FEC = 0.0
c         FAC = 0.0
c         FPC = 0.0
c        END IF
c
c. . . . . . . . . . . . . . . . . . . . . . .
c
c
c    P U L S E D   D I T H E R   S I G N A L
c
c                  SUBLIMINAL
c. . . . . . . . . . . . . . . . . . . . . . .
c

c       Don=amod(t,3.0)

c       IF((Don.ge.0.0).and.(Don.lt.0.125))THEN
c         FEC = 15.2
c         FAC = -7.0
c         FPC = 22.0
c        ELSE IF((Don.ge.0.125).and.(Don.lt.0.25))THEN
c         FEC =-16.5
c         FAC =  8.0
c         FPC = 22.0
c        ELSE
c         FEC = 0.0
c         FAC = 0.0
c         FPC = 0.0
c        END IF

c
c. . . . . . . . . . . . . . . . . . . . . . . . . . . . .
c
c    S I N U S O I D A L   D I T H E R   S I G N A L
c
c. . . . . . . . . . . . . . . . . . . . . . . . . . . . .
c

c    FREQUENCY

        omega1 = 15.0
        omega2 = 15.0
        omega3 = 15.0
```

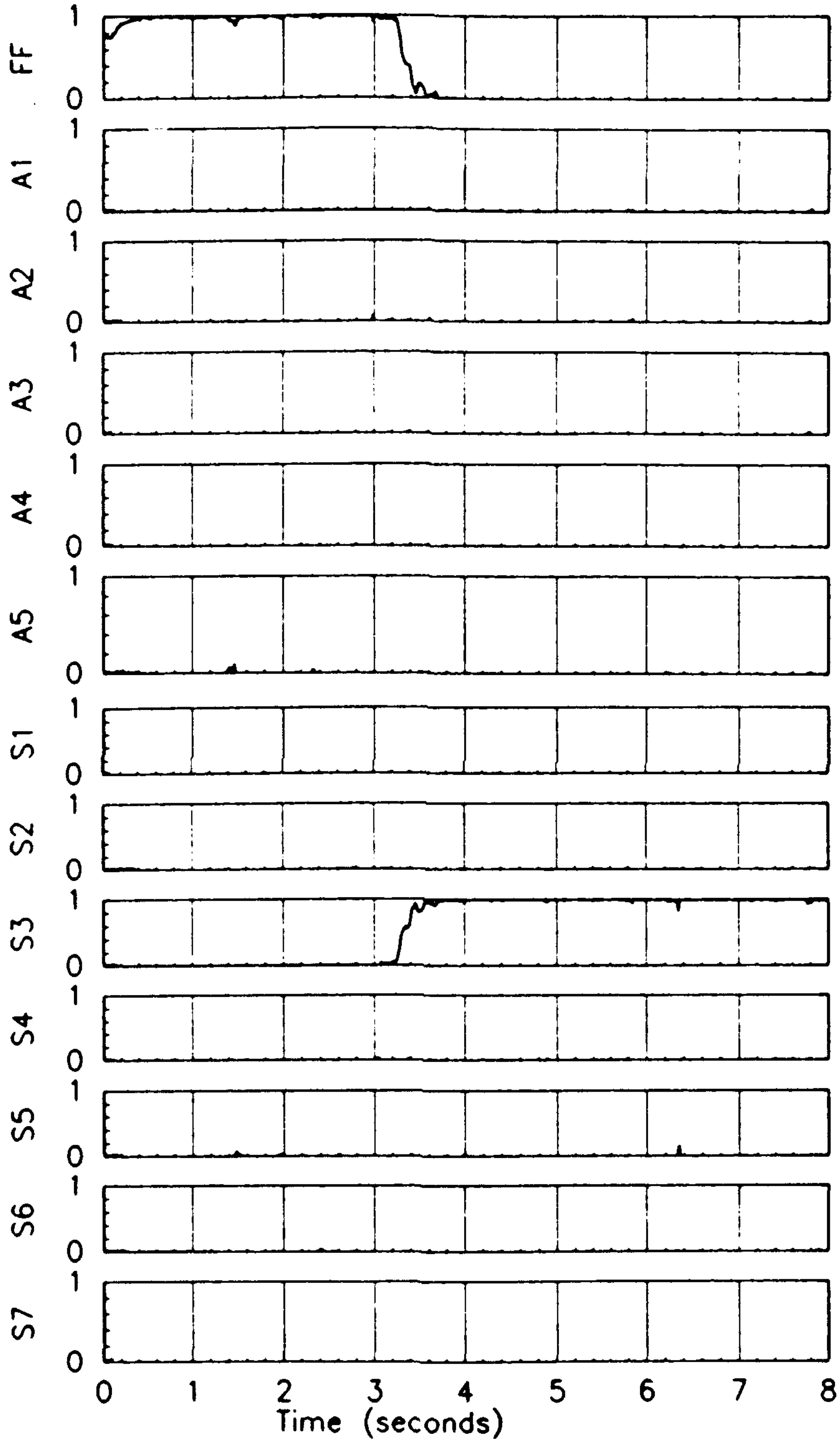
```
c    SIGNAL

         FBC= 12.0*sin(omega1*t)
         IF(FBC.LT.0.0)FBC=12.5*sin(omega1*t)
         FAC= -11.0*sin(omega2*t)
         FPC= 30.0*sin(omega3*t)

c  . . . . . . . . . . . . . . . . . . . . . . . . . . . .
c
c
c  U S E R   D E F I N E D   D I T H E R   S I G N A L
c
c
c  . . . . . . . . . . . . . . . . . . . . . . . . . . . .

c      Don=amod(t,3.0)

c      IF((Don.ge.0.0).and.(Don.lt.0.1))THEN
c        FBC = 15.2
c        FAC = 16.0
c        FPC = 55.0
c      ELSE IF((Don.ge.0.1).and.(Don.lt.0.125))THEN
c        FBC = (-15.2/0.025)*(Don - 0.1) + 15.2
c        FAC = (-16.0/0.025)*(Don - 0.1) + 16.0
c        FPC = (-55.0/0.025)*(Don - 0.1) + 55.0
c      ELSE IF((Don.ge.0.125).and.(Don.lt.0.3))THEN
c        FBC =-16.5
c        FAC =-14.0
c        FPC =-50.0
c      ELSE IF((Don.ge.0.3).and.(Don.lt.0.325))THEN
c        FBC = (16.5/0.025)*(Don - 0.3) - 16.5
c        FAC = (14.0/0.025)*(Don - 0.3) - 14.0
c        FPC = (50.0/0.025)*(Don - 0.3) - 50.0
c      ELSE
c        FBC = 0.0
c        FAC = 0.0
c        FPC = 0.0
c      END IF

c . . . . . . . . . . . . . . . . . . . . . . . . . . . .
c
c
c P I L O T   P U R P O S E F U L   C O M M A N D S
c
c
c . . . . . . . . . . . . . . . . . . . . . . . . . . . .

c     Pitch Path

c       IF((t.GT.2.95).and.(t.LT.4.45))THEN
c         FAC=13.5
c        FAC=20.0
c        FPC=-40.0
c       ENDIF

c     Roll Path


c     Yaw Path


      RETURN
      END
```



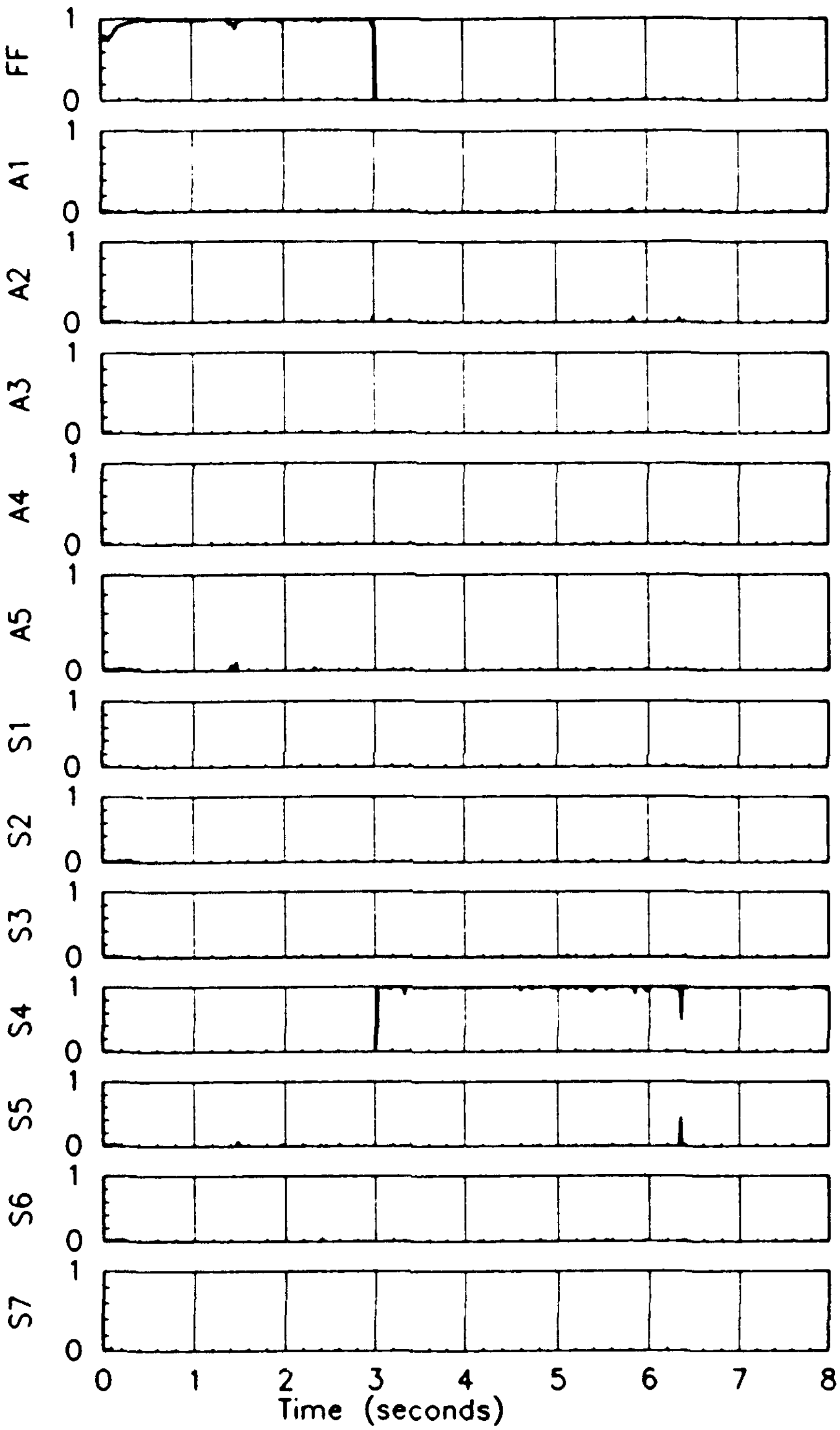
A.7 Probabilities for a velocity sensor failure using a sinusoidal dither signal



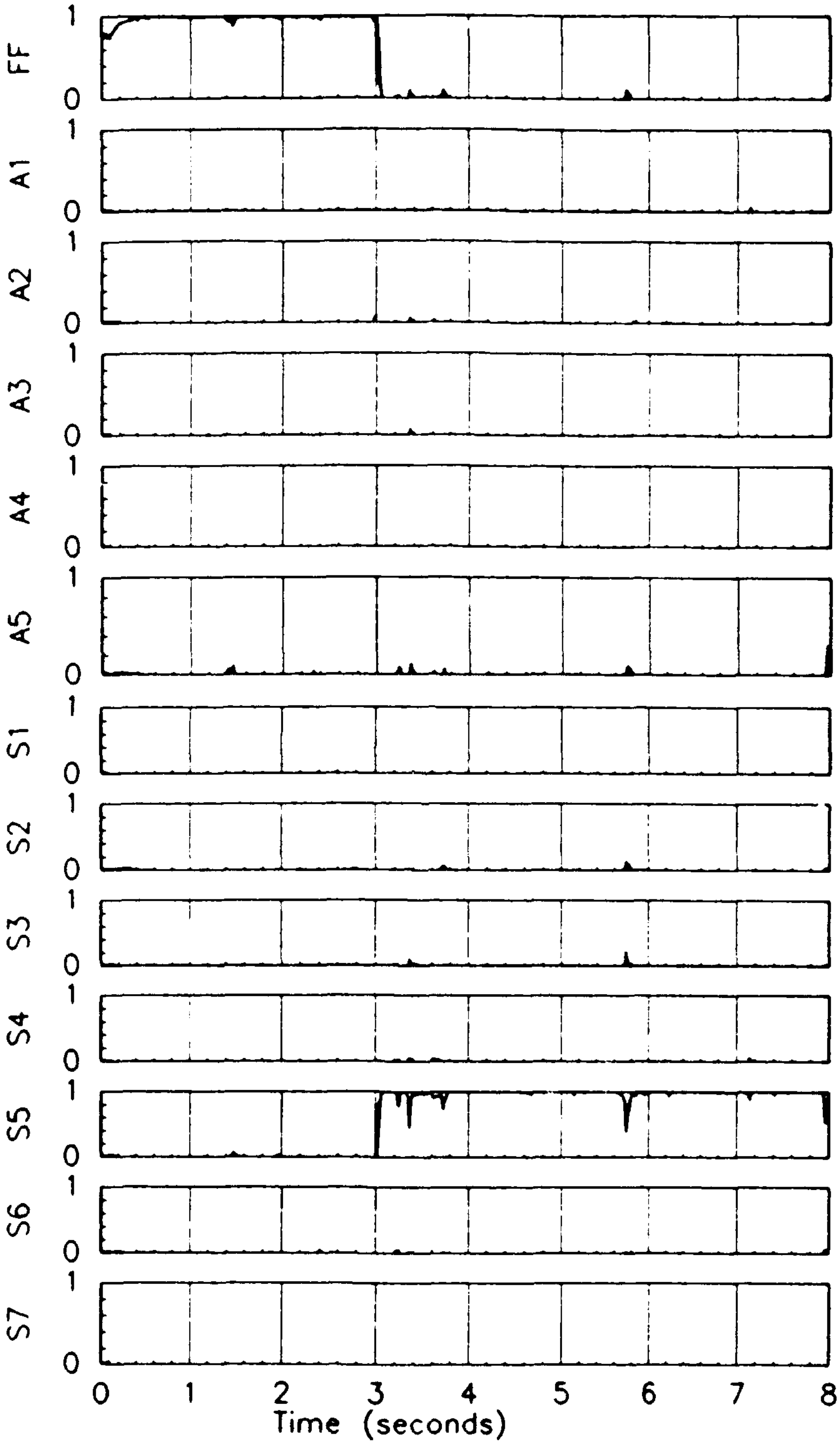
A.8 Probabilities for a angle of attack sensor failure using a sinusoidal dither signal



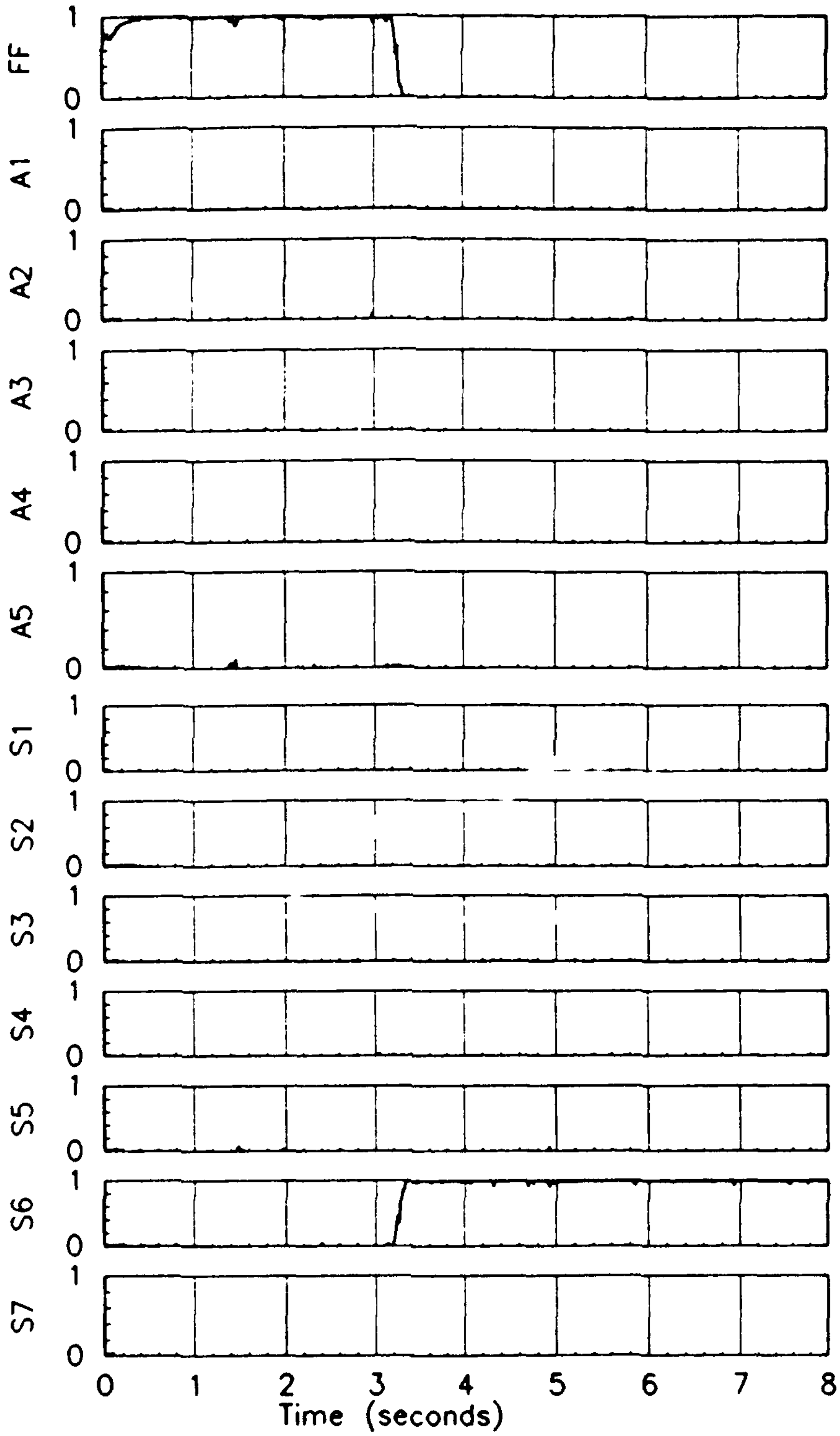
A.9 Probabilities for a pitch rate sensor failure using a sinusoidal dither signal



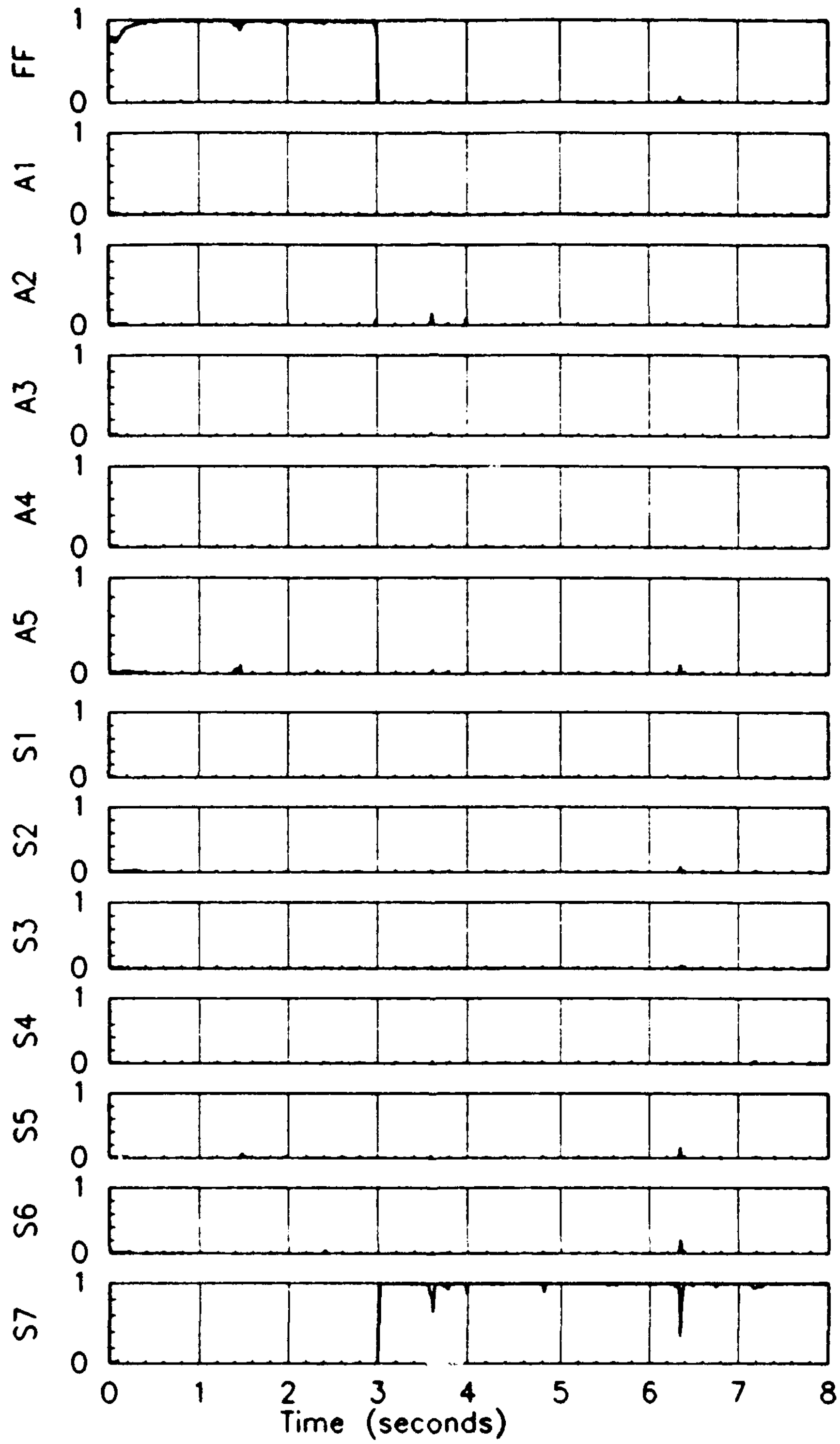
A.10 Probabilities for a normal acceleration sensor failure using a sinusoidal dither signal



A.11 Probabilities for a roll rate sensor failure using a sinusoidal dither signal



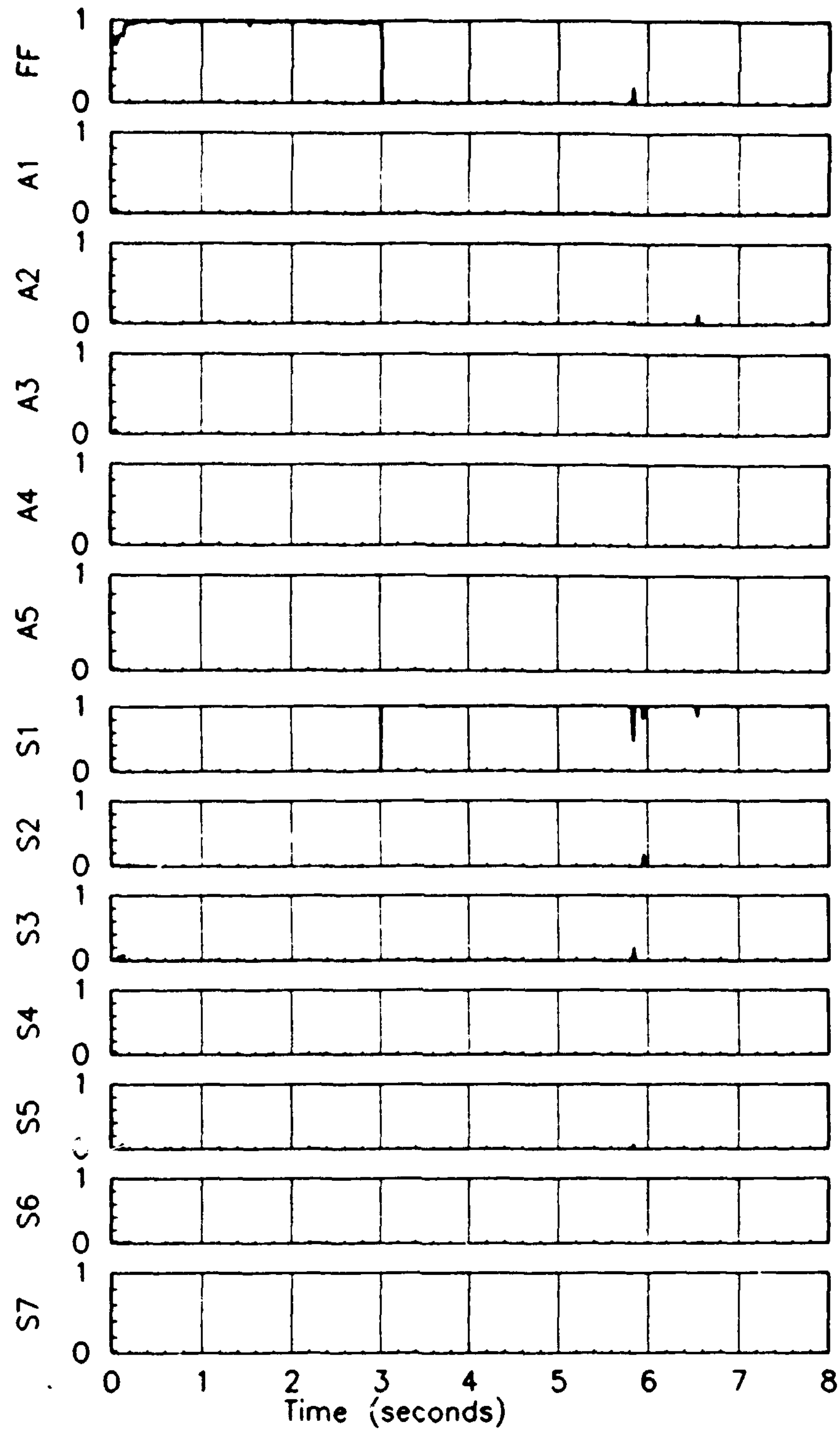
A.12 Probabilities for a yaw rate sensor failure using a sinusoidal dither signal



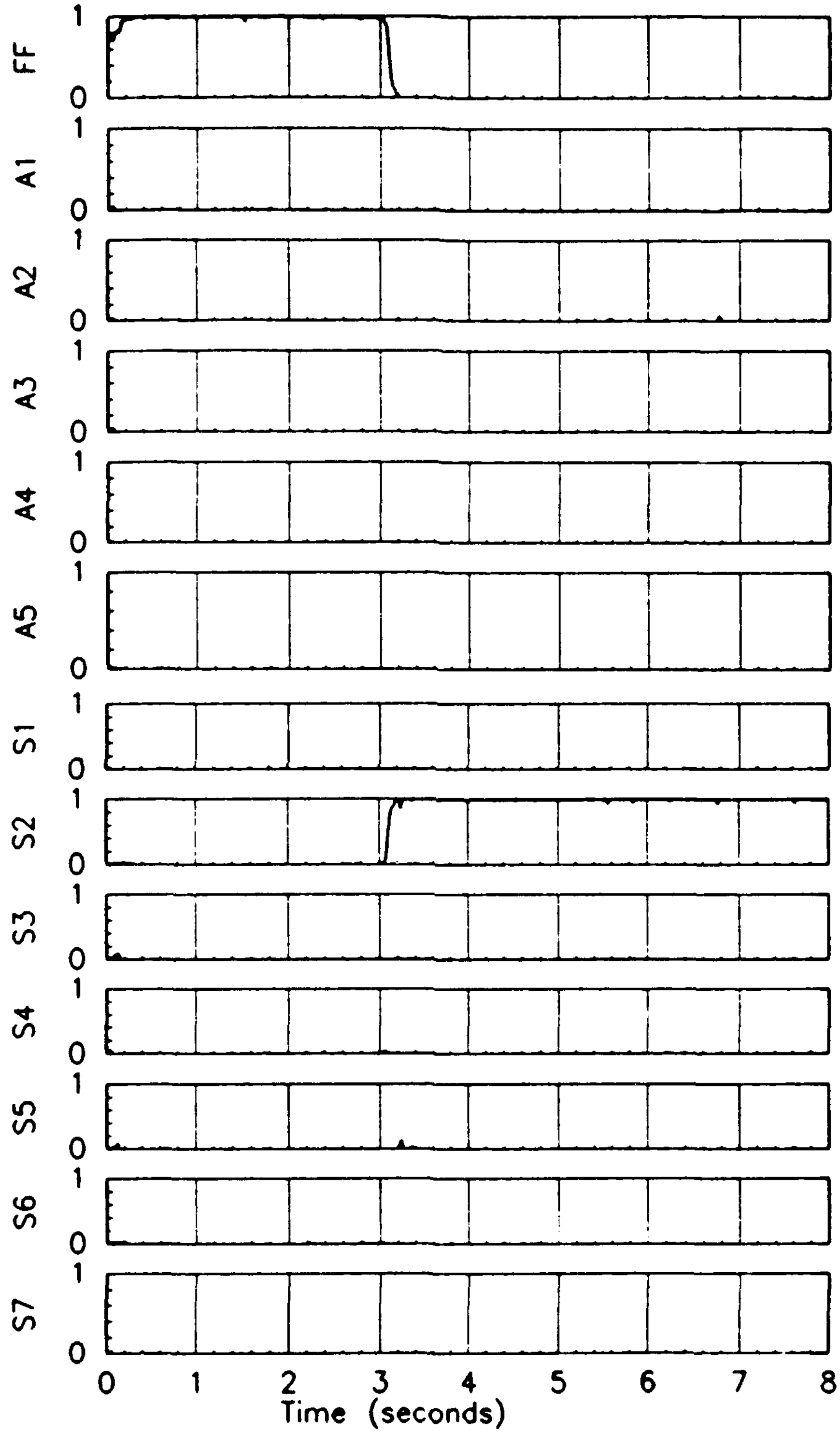
A.13 Probabilities for a lateral acceleration sensor failure using a sinusoidal dither signal

```
      SUBROUTINE COMMAND(FBC,FAC,FPC,T)
c --- Provides the vista flight control system (CNTRL.FOR)
c     with the command signals for the longitudinal axis
c     (FBC), the lateral command (FBC), and the directional
c     command (FPC).  Additionally, simulation results have
c     indicated the need for a dither signal to "shake up"
c     the system and aid the filters in their identification
c     tasks.

      INCLUDE 'DECLARR.TXT'
      REAL FBC,FAC,FPC,DON,T,omega1,omega2,omega3


      SAVE

c     COMMAND SIGNALS


c . . . . . . . . . . . . . . . . . . . . . . . .
c  Dither signal generation
c    --NOTE:  if a control command is to be used,
c             it must be added to the below
c             created dither signal.
c . . . . . . . . . . . . . . . . . . . . . . . .
c
c
c   P U L S E D   D I T H E R   S I G N A L
c
c              NON - SUBLIMINAL
c . . . . . . . . . . . . . . . . . . . . . . . .

c       Don=amod(t,3.0)

c       IF((Don.ge.0.0).and.(Don.lt.0.125))THEN
c         FBC = 13.5
c         FAC = -7.5
c         FPC = 24.0
c        ELSE IF((Don.ge.0.125).and.(Don.lt.0.25))THEN
c          FBC =-13.0
c          FAC = 7.5
c          FPC =-24.0
c        ELSE
c          FBC = 0.0
c          FAC = 0.0
c          FPC = 0.0
c        END IF
c
c . . . . . . . . . . . . . . . . . . . . . . . .
c
c
c   P U L S E D   D I T H E R   S I G N A L
c
c                SUBLIMINAL
c . . . . . . . . . . . . . . . . . . . . . . . .
c

c       Don=amod(t,3.0)

c       IF((Don.ge.0.0).and.(Don.lt.0.125))THEN
c          FBC = 15.2
c          FAC = -7.0
c          FPC = 22.0
c        ELSE IF((Don.ge.0.125).and.(Don.lt.0.25))THEN
c          FBC =-16.5
c          FAC =  8.0
c          FPC = 22.0
c        ELSE
c          FBC = 0.0
c          FAC = 0.0
c          FPC = 0.0
c        END IF

c
c . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
c
c    S I N U S O I D A L   D I T H E R   S I G N A L
c
c . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
c

c     FREQUENCY

        omega1 = 15.0
        omega2 = 15.0
        omega3 = 15.0
```

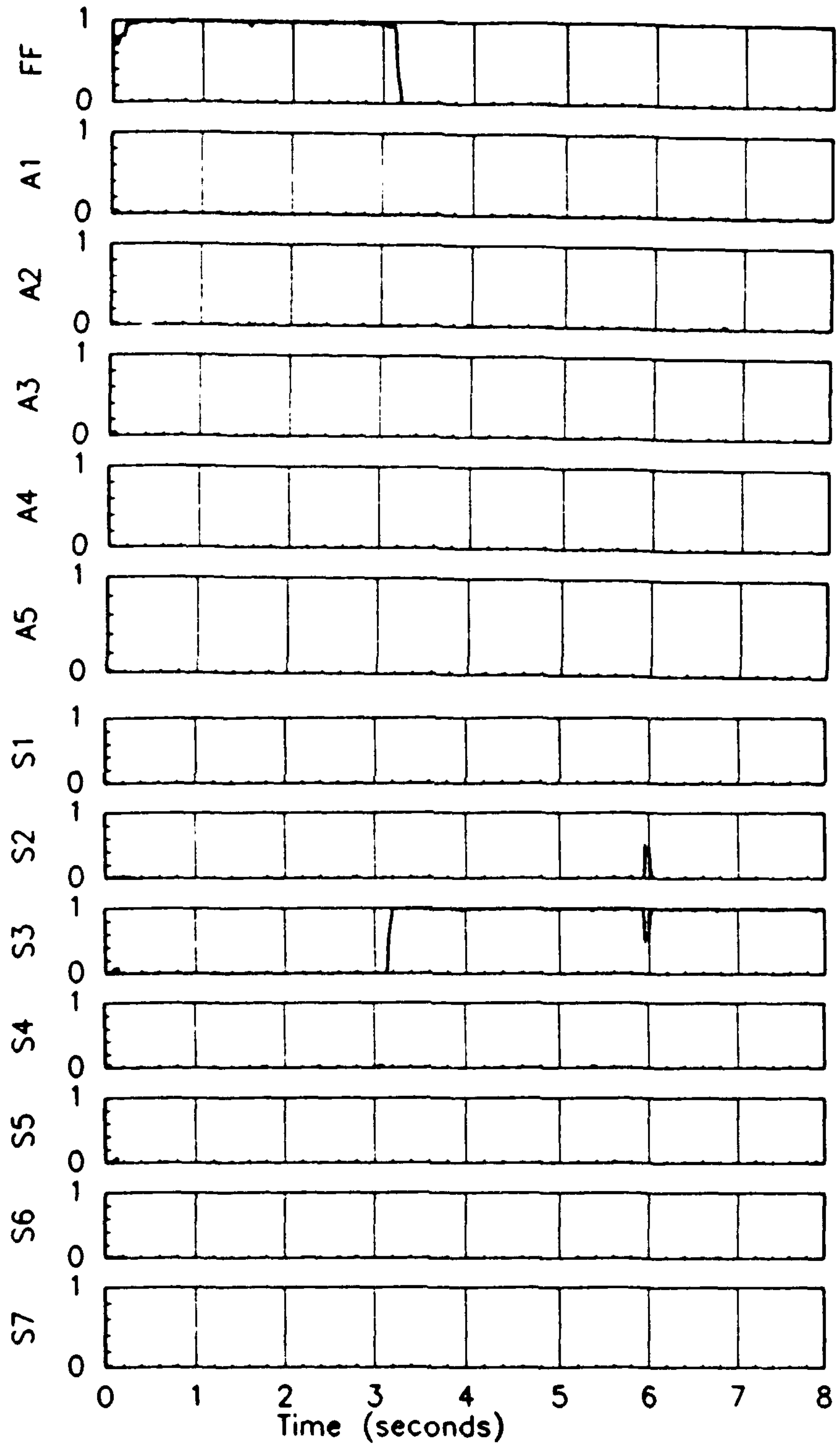
```
c     SIGNAL

          FEC= 12.0*sin(omega1*t)
          IF(FEC.LT.0.0)FEC=12.5*sin(omega1*t)
          FAC= -11.0*sin(omega2*t)
          FPC= 30.0*sin(omega3*t)

c   . . . . . . . . . . . . . . . . . . . . . . . . . . .
c
c
c   U S E R   D E F I N E D   D I T H E R   S I G N A L
c
c
c   . . . . . . . . . . . . . . . . . . . . . . . . . . .

c      Don=amod(t,3.0)

c      IF((Don.ge.0.0).and.(Don.lt.0.1))THEN
c        FEC = 15.2
c        FAC = 16.0
c        FPC = 55.0
c      ELSE IF((Don.ge.0.1).and.(Don.lt.0.125))THEN
c        FEC = (-15.2/0.025)*(Don - 0.1) + 15.2
c        FAC = (-16.0/0.025)*(Don - 0.1) + 16.0
c        FPC = (-55.0/0.025)*(Don - 0.1) + 55.0
c      ELSE IF((Don.ge.0.125).and.(Don.lt.0.3))THEN
c        FEC =-16.5
c        FAC =-14.0
c        FPC =-50.0
c      ELSE IF((Don.ge.0.3).and.(Don.lt.0.325))THEN
c        FEC = (16.5/0.025)*(Don - 0.3) - 16.5
c        FAC = (14.0/0.025)*(Don - 0.3) - 14.0
c        FPC = (50.0/0.025)*(Don - 0.3) - 50.0
c      ELSE
c        FEC = 0.0
c        FAC = 0.0
c        FPC = 0.0
c      END IF

c . . . . . . . . . . . . . . . . . . . . . . . . . . .
c
c
c P I L O T   P U R P O S E F U L   C O M M A N D S
c
c
c . . . . . . . . . . . . . . . . . . . . . . . . . . .

c     Pitch Path

c       IF((t.GT.2.95).and.(t.LT.4.45))THEN
c         FAC=13.5
         FAC=20.0
c        FPC=-40.0
c       ENDIF

c     Roll Path


c     Yaw Path

      RETURN
      END
```



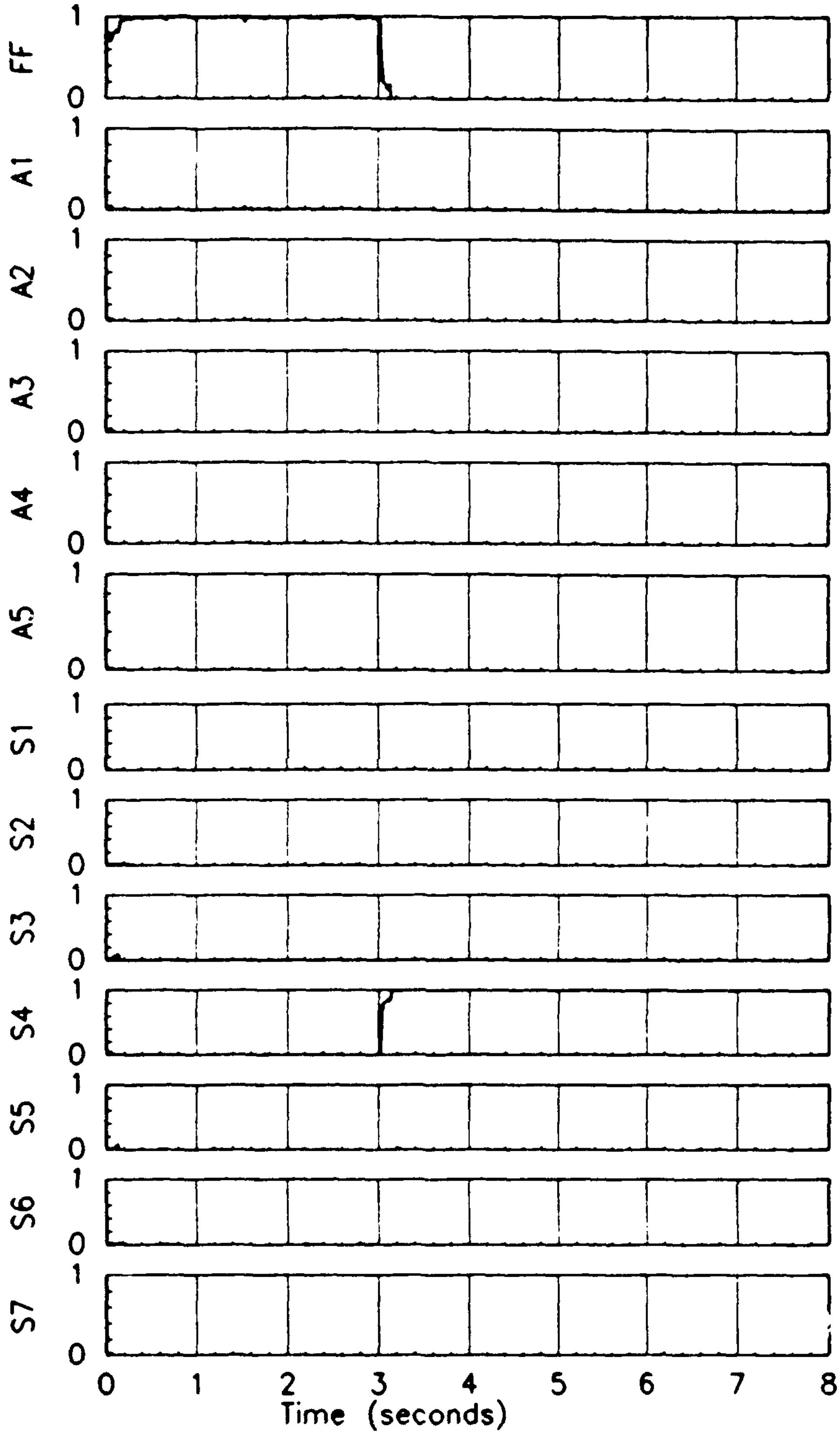
A.14 Probabilities for a velocity sensor failure using a purposeful roll command



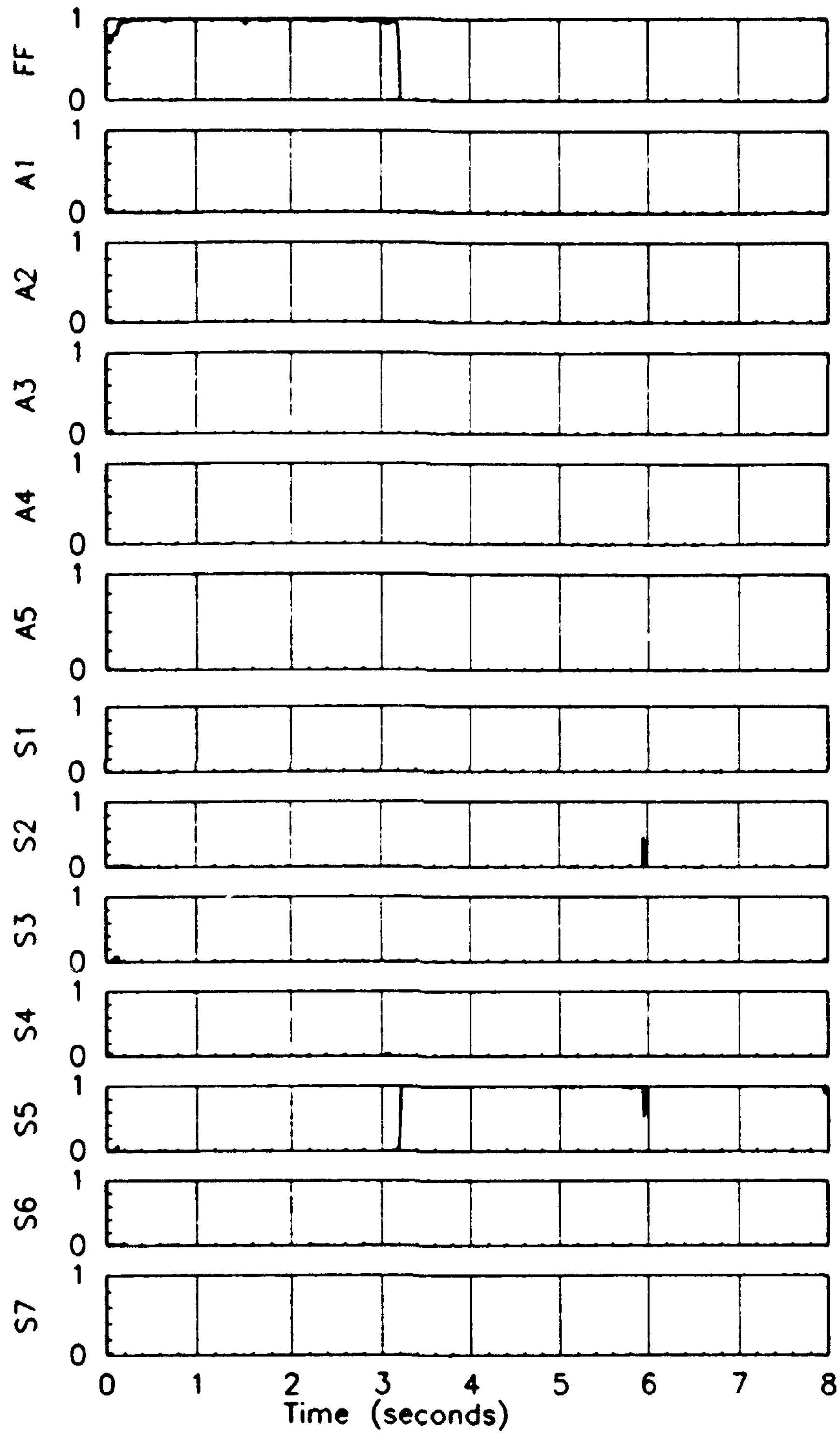
A.15 Probabilities for a angle of attack sensor failure using a purposeful roll command



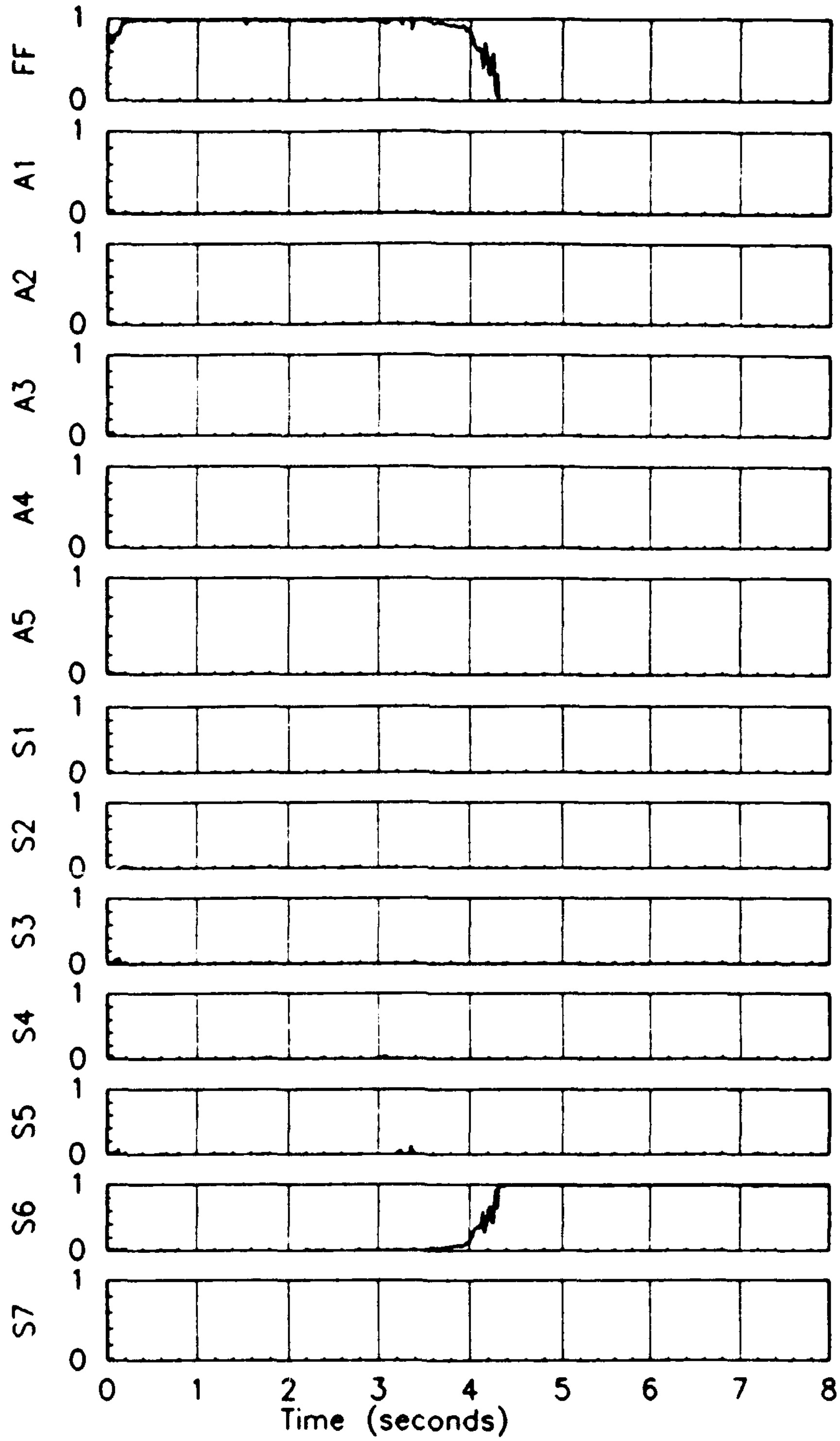
A.16 Probabilities for a pitch rate sensor failure using a purposeful roll command



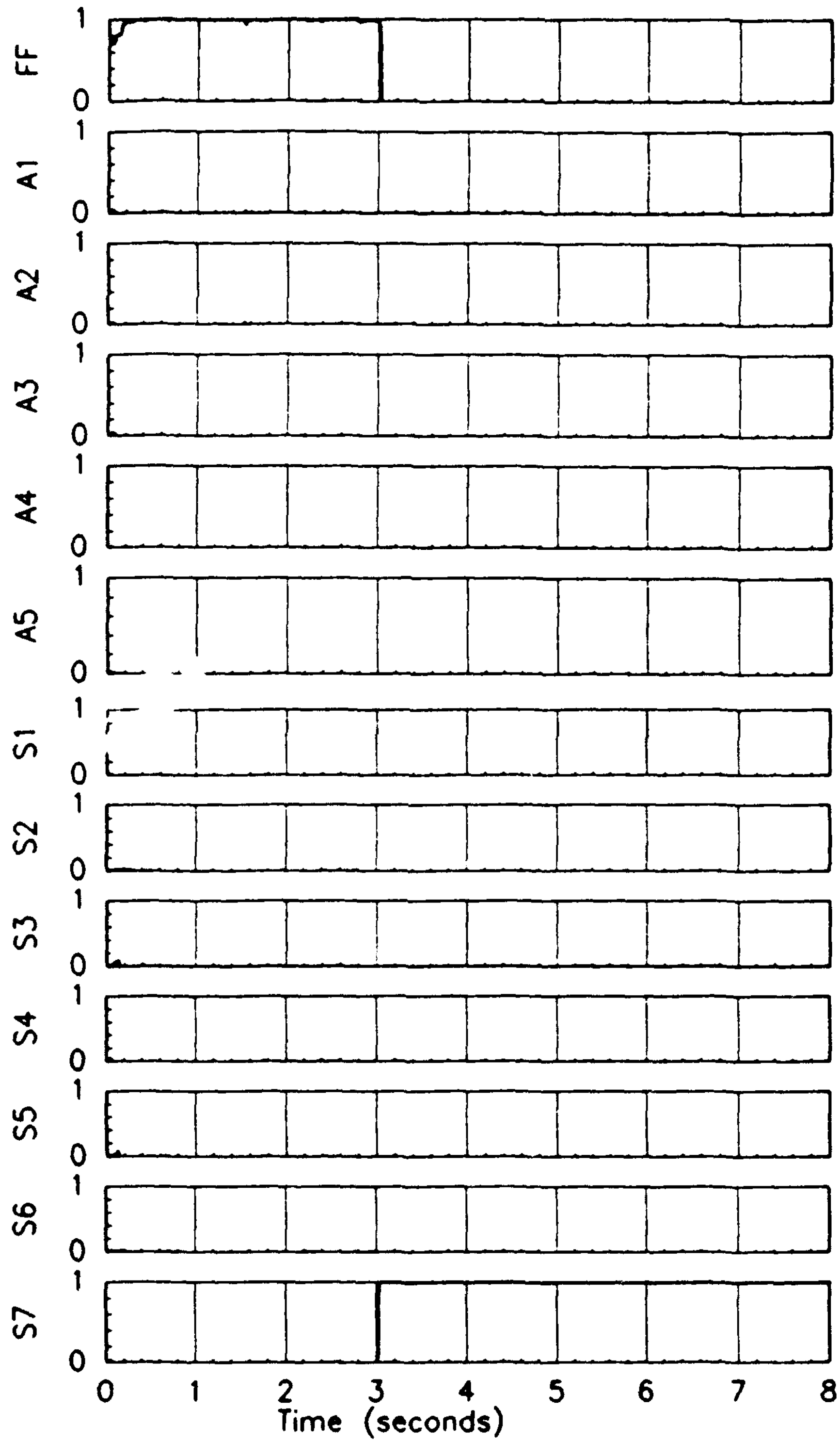
A.17 Probabilities for a normal acceleration sensor failure using a purposeful roll command



A.18 Probabilities for a roll rate sensor failure using a purposeful roll command



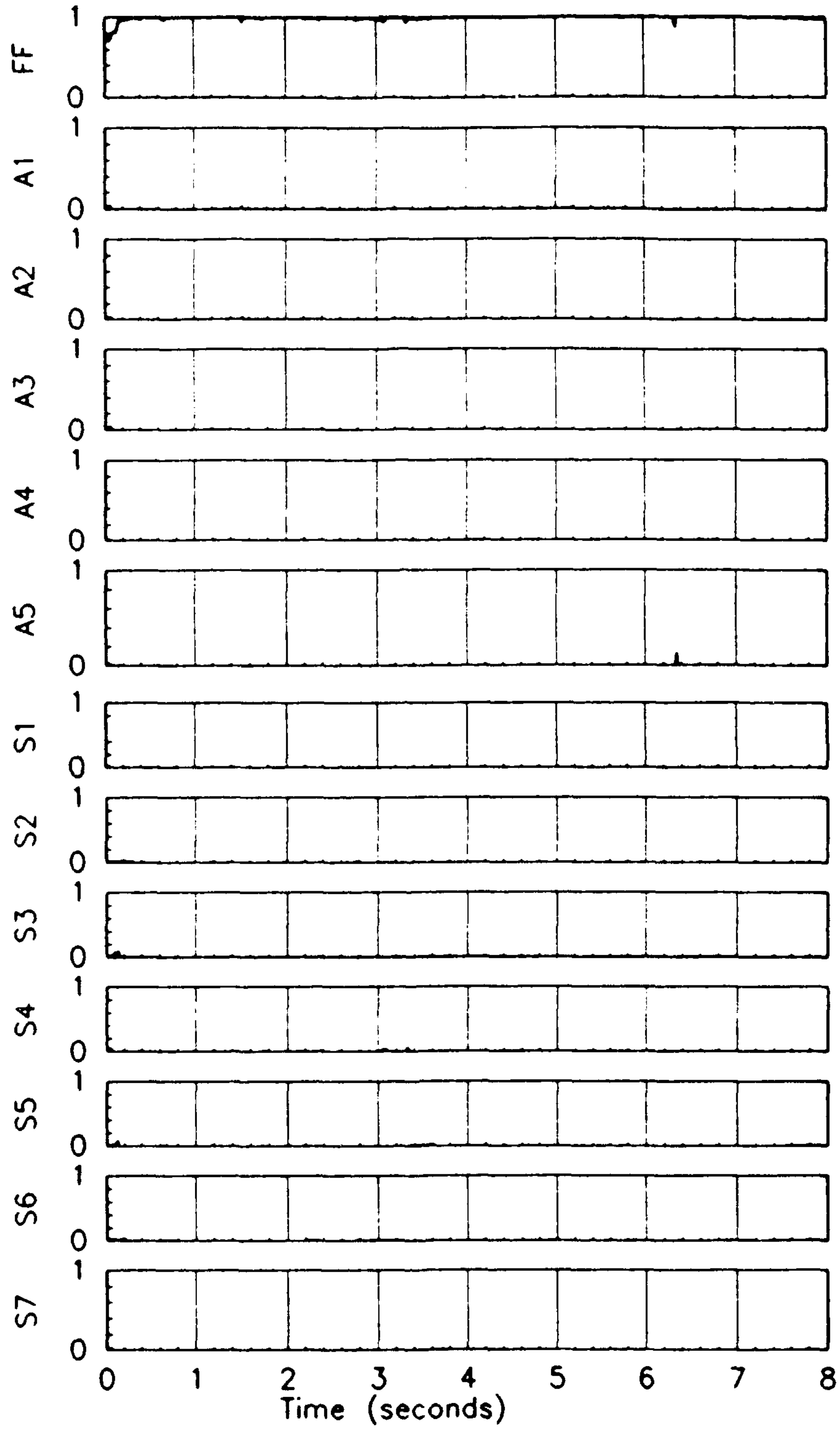
A.19 Probabilities for a yaw rate sensor failure using a purposeful roll command



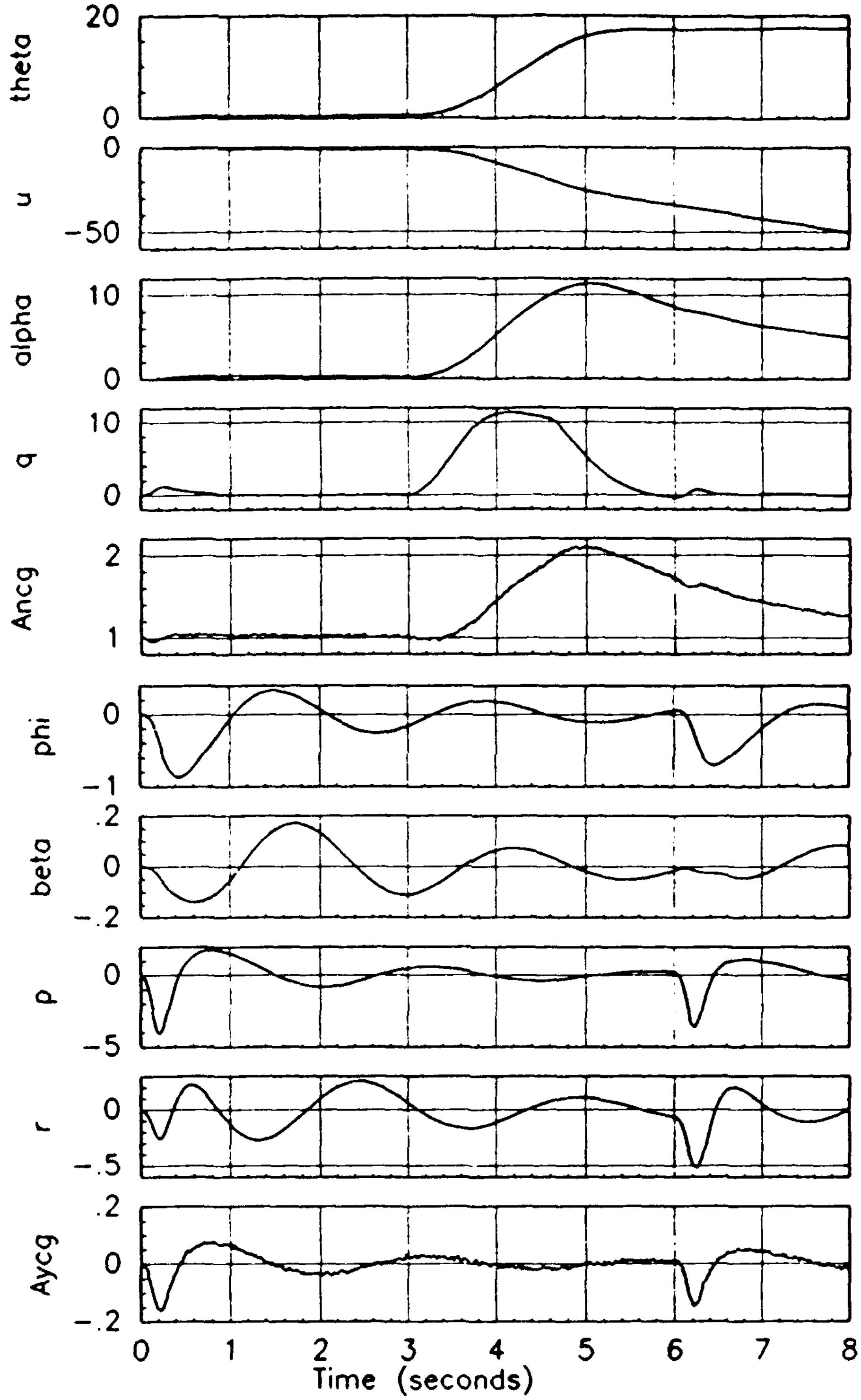
A.20 Probabilities for a lateral acceleration sensor failure using a purposeful roll command

```
      SUBROUTINE COMMAND(FBC,FAC,FPC,T)
c ---- Provides the vista flight control system [CNTRL.FOR]
c      with the command signals for the longitudinal axis
c      [FBC], the lateral command [FBC], and the directional
c      command [FPC].  Additionally, simulation results have
c      indicated the need for a dither signal to "shake up"
c      the system and aid the filters in their identification
c      tasks.

      INCLUDE 'DECLARR.TXT'
      REAL FBC,FAC,FPC,DON,T,omega1,omega2,omega3


      SAVE

c     COMMAND SIGNALS


c . . . . . . . . . . . . . . . . . . . . . .
c  Dither signal generation
c    --NOTE:  if a control command is to be used,
c             it must be added to the below
c             created dither signal.
c . . . . . . . . . . . . . . . . . . . . . .
c
c
c    P U L S E D   D I T H E R   S I G N A L
c
c             NON - SUBLIMINAL
c . . . . . . . . . . . . . . . . . . . . . .

        Don=amod(t,3.0)

        IF((Don.ge.0.0).and.(Don.lt.0.125))THEN
          FBC = 13.5
          FAC = -7.5
          FPC = 24.0
         ELSE IF((Don.ge.0.125).and.(Don.lt.0.25))THEN
           FBC =-13.0
           FAC = 7.5
           FPC =-24.0
         ELSE
           FBC = 0.0
           FAC = 0.0
           FPC = 0.0
        END IF
c
c . . . . . . . . . . . . . . . . . . . . . .
c
c
c    P U L S E D   D I T H E R   S I G N A L
c
c                 SUBLIMINAL
c . . . . . . . . . . . . . . . . . . . . . .
c

c       Don=amod(t,3.0)

c       IF((Don.ge.0.0).and.(Don.lt.0.125))THEN
c          FBC = 15.2
c          FAC = -7.0
c          FPC = 22.0
c        ELSE IF((Don.ge.0.125).and.(Don.lt.0.25))THEN
c          FBC =-16.5
c          FAC =  8.0
c          FPC = 22.0
c        ELSE
c          FBC = 0.0
c          FAC = 0.0
c          FPC = 0.0
c        END IF

c
c . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
c
c     S I N U S O I D A L   D I T H E R   S I G N A L
c
c . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
c

c   FREQUENCY

       omega1 = 15.0
       omega2 = 15.0
       omega3 = 15.0
```

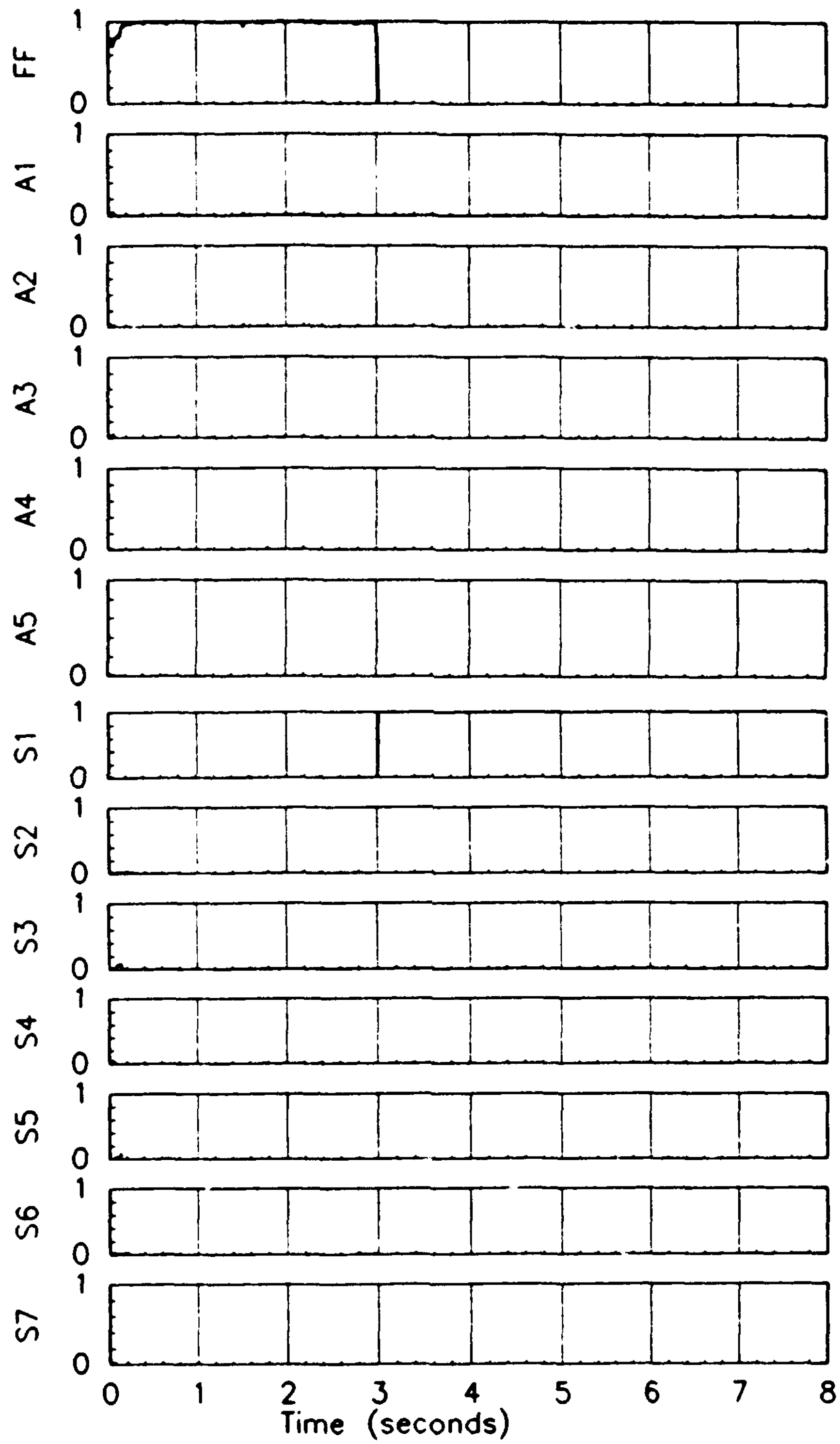
```
c     SIGNAL
c           FBC= 12.0*sin(omega1*t)
c           IF(FBC.LT.0.0)FBC=12.5*sin(omega1*t)
c           FAC= -11.0*sin(omega2*t)
c           FPC= 30.0*sin(omega3*t)
c  . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
c
c
c  U S E R   D E F I N E D   D I T H E R   S I G N A L
c
c
c  . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
c       Don=amod(t,3.0)
c       IF((Don.ge.0.0).and.(Don.lt.0.1))THEN
c         FBC = 15.2
c         FAC = 16.0
c         FPC = 55.0
c       ELSE IF((Don.ge.0.1).and.(Don.lt.0.125))THEN
c         FBC = (-15.2/0.025)*(Don - 0.1) + 15.2
c         FAC = (-16.0/0.025)*(Don - 0.1) + 16.0
c         FPC = (-55.0/0.025)*(Don - 0.1) + 55.0
c       ELSE IF((Don.ge.0.125).and.(Don.lt.0.3))THEN
c         FBC =-16.5
c         FAC =-14.0
c         FPC =-50.0
c       ELSE IF((Don.ge.0.3).and.(Don.lt.0.325))THEN
c         FBC = (16.5/0.025)*(Don - 0.3) - 16.5
c         FAC = (14.0/0.025)*(Don - 0.3) - 14.0
c         FPC = (50.0/0.025)*(Don - 0.3) - 50.0
c       ELSE
c         FBC = 0.0
c         FAC = 0.0
c         FPC = 0.0
c       END IF
c  . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
c
c
c  P I L O T   P U R P O S E F U L   C O M M A N D S
c
c
c  . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
c       Pitch Path
        IF((t.GT.2.95).and.(t.LT.3.15))THEN
c           FBC=13.5
          FAC=20.0
c         FPC=-40.0
        ENDIF
        IF((T.GT.3.15).and.(t.LT.4.55))THEN
          FBC=13.5
          FAC=10.0
        ENDIF
c       Roll Path

c       Yaw Path

        RETURN
        END
```



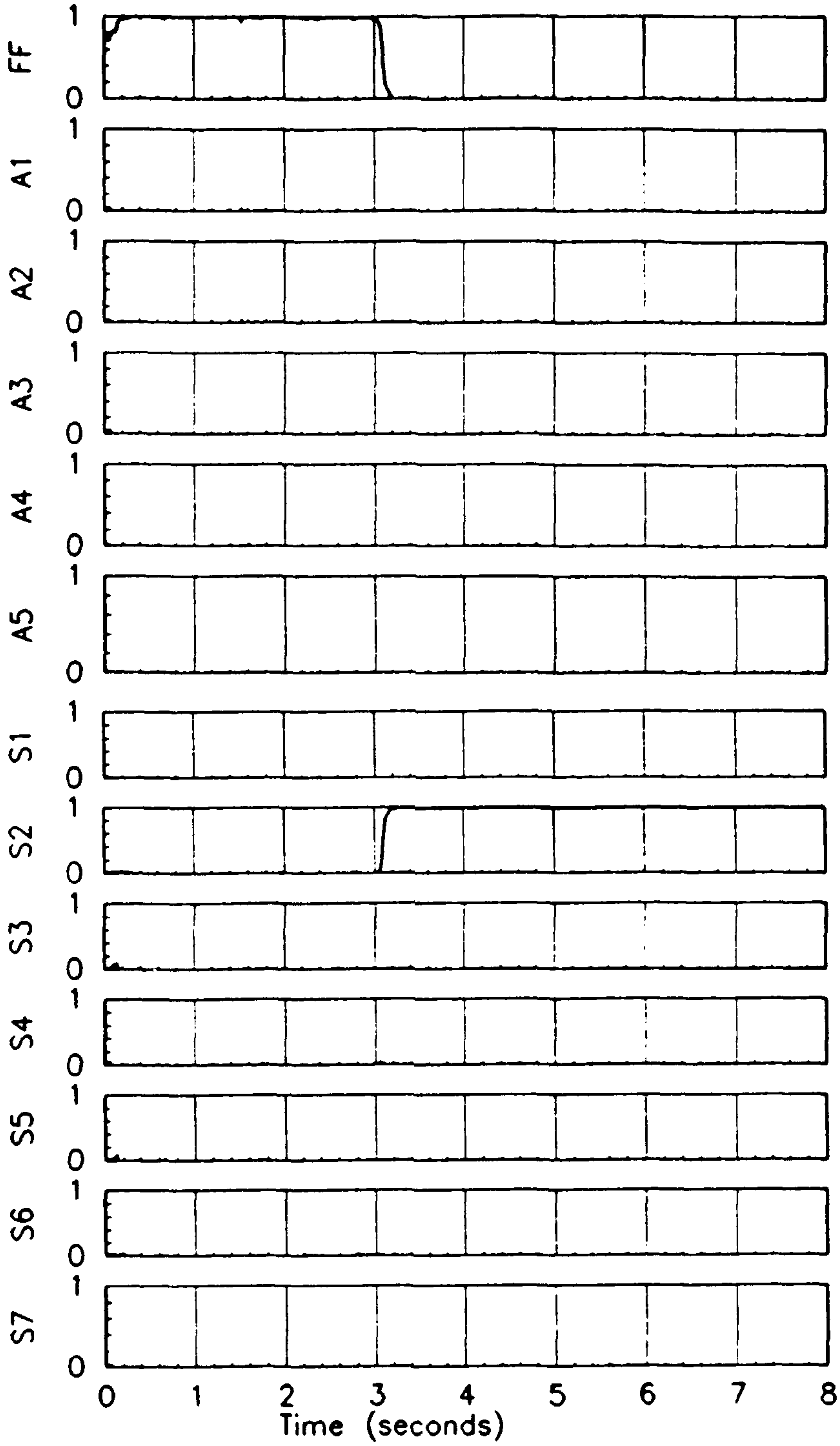
A.21 Probabilities for a no-failure scenario using a purposeful roll and pull command



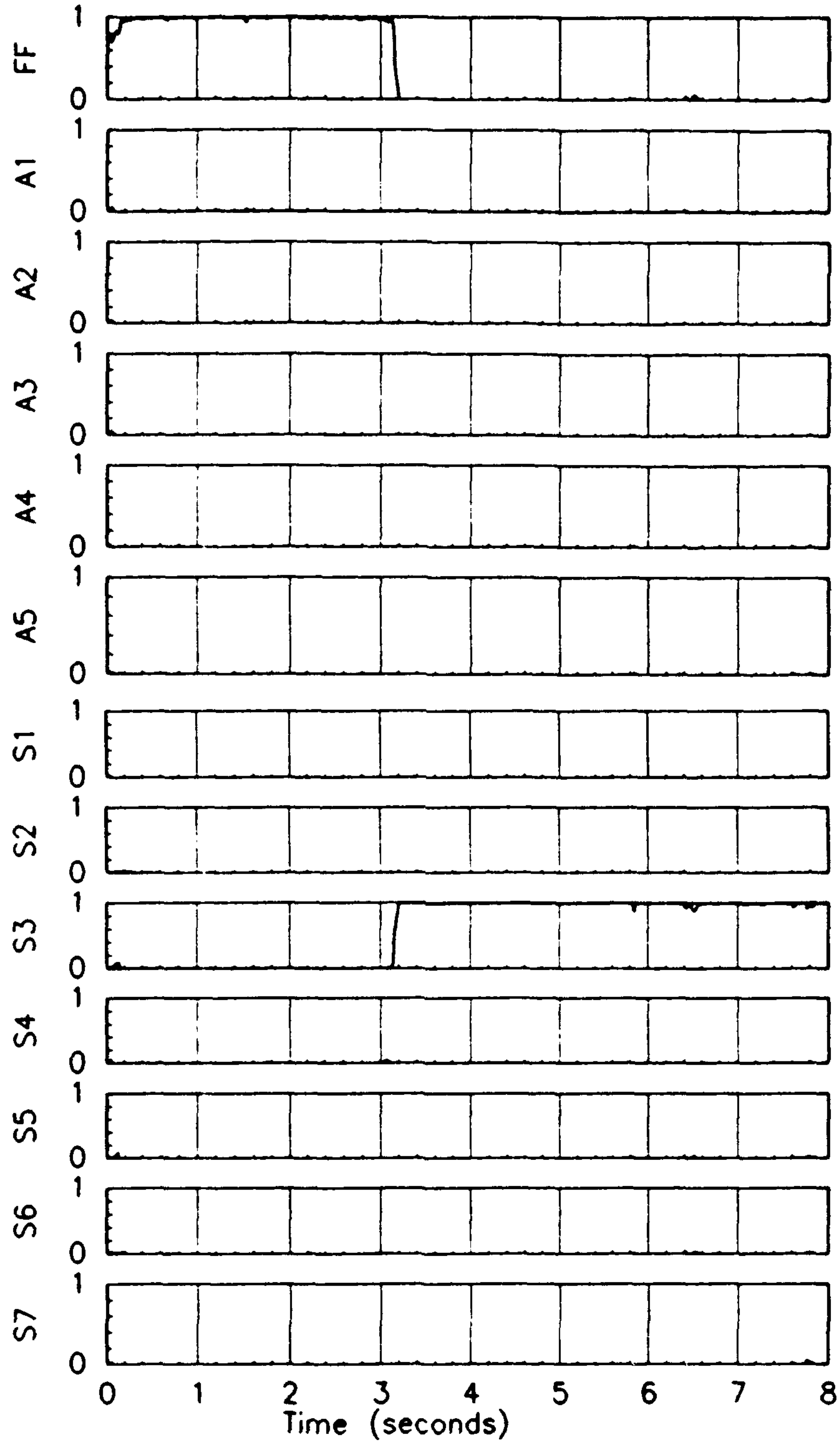
A.22 States for a no-failure scenario using a purposeful roll and pull command



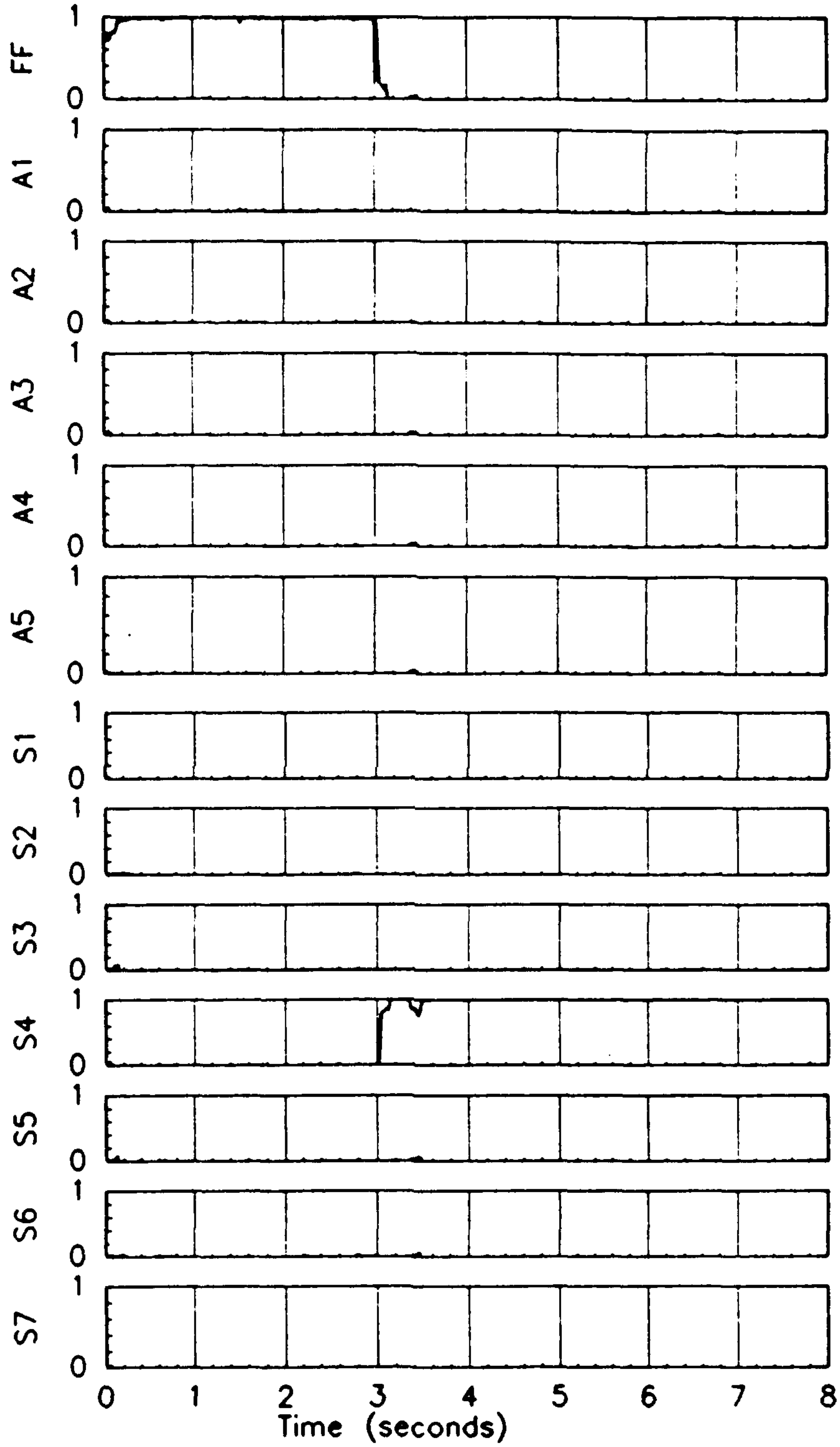
A.23 Probabilities for a velocity sensor failure using a purposeful roll and pull command



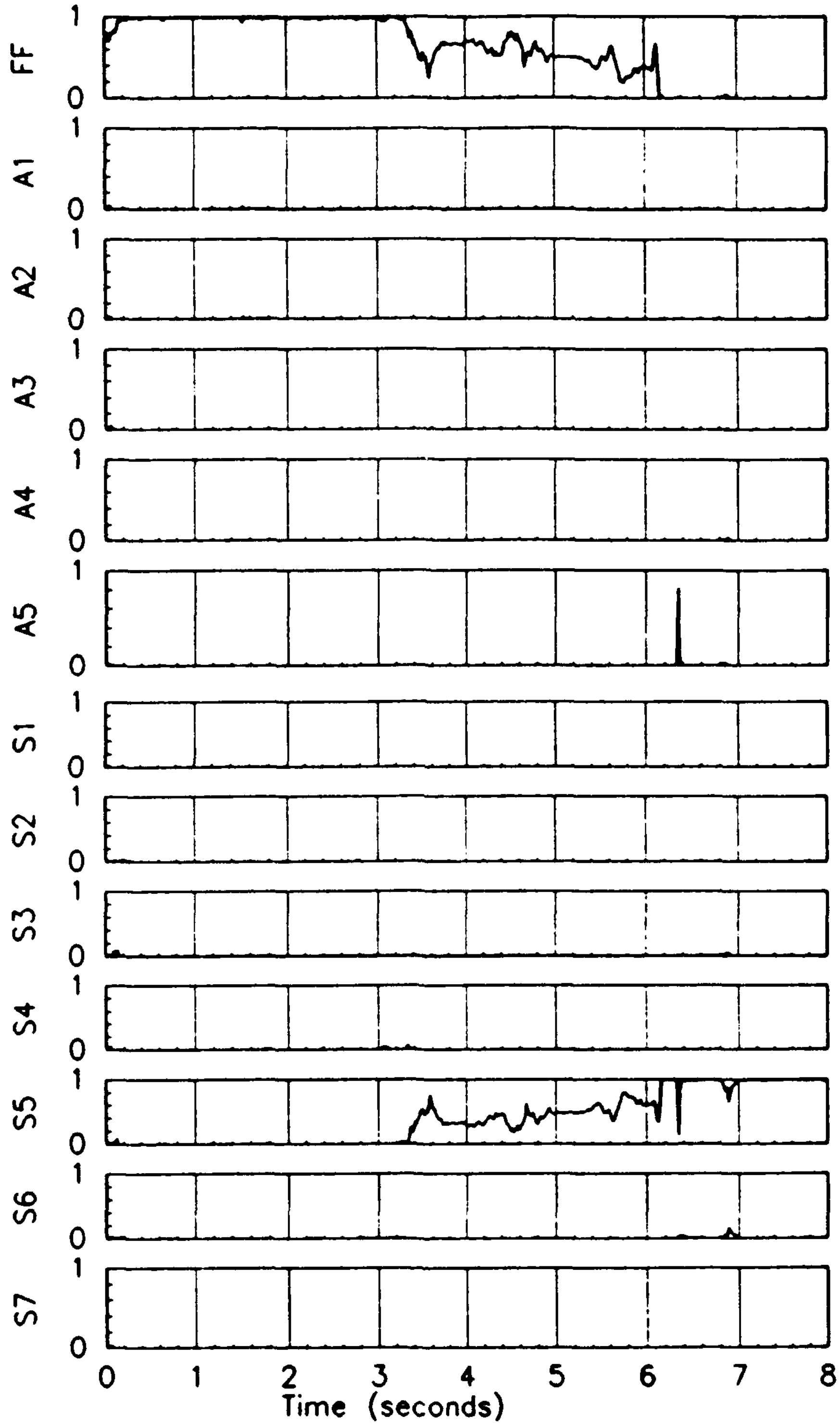
A.24 Probabilities for a angle of attack sensor failure using a purposeful roll and pull command



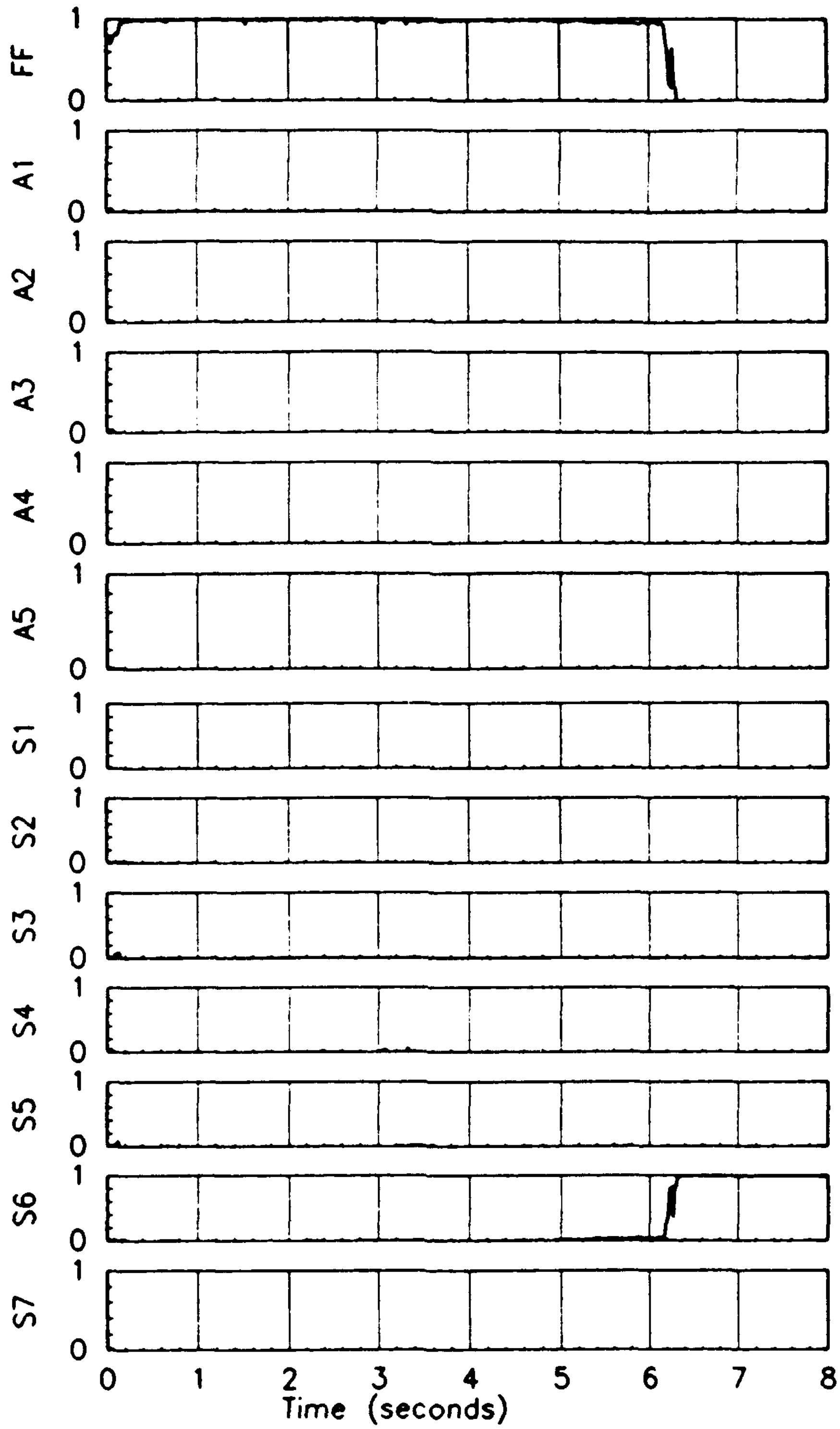
A.25 Probabilities for a pitch rate sensor failure using a purposeful roll and pull command



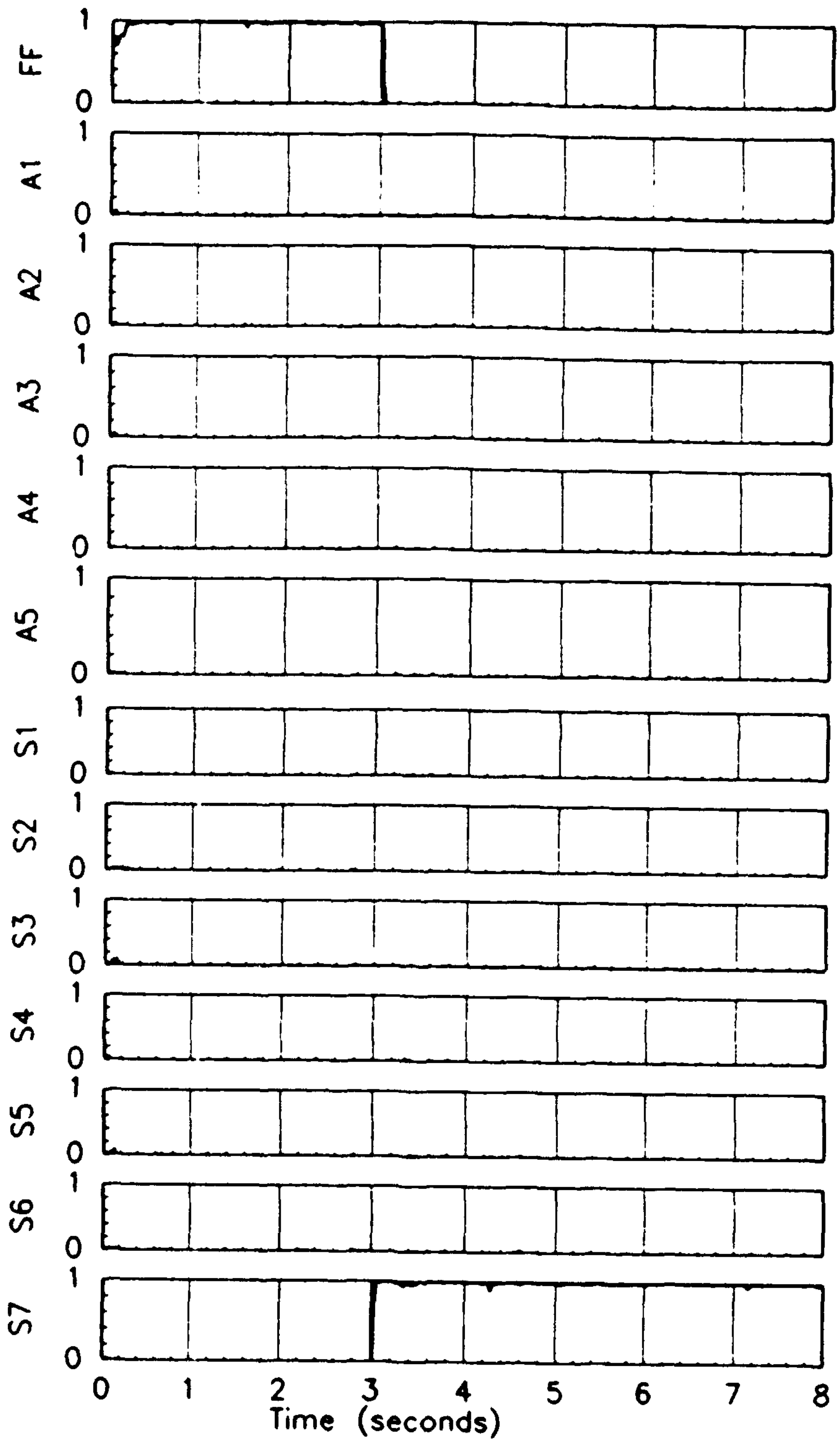
A.26 Probabilities for a normal acceleration sensor failure using a purposeful roll and pull command



A.27 Probabilities for a roll rate sensor failure using a purposeful roll and pull command



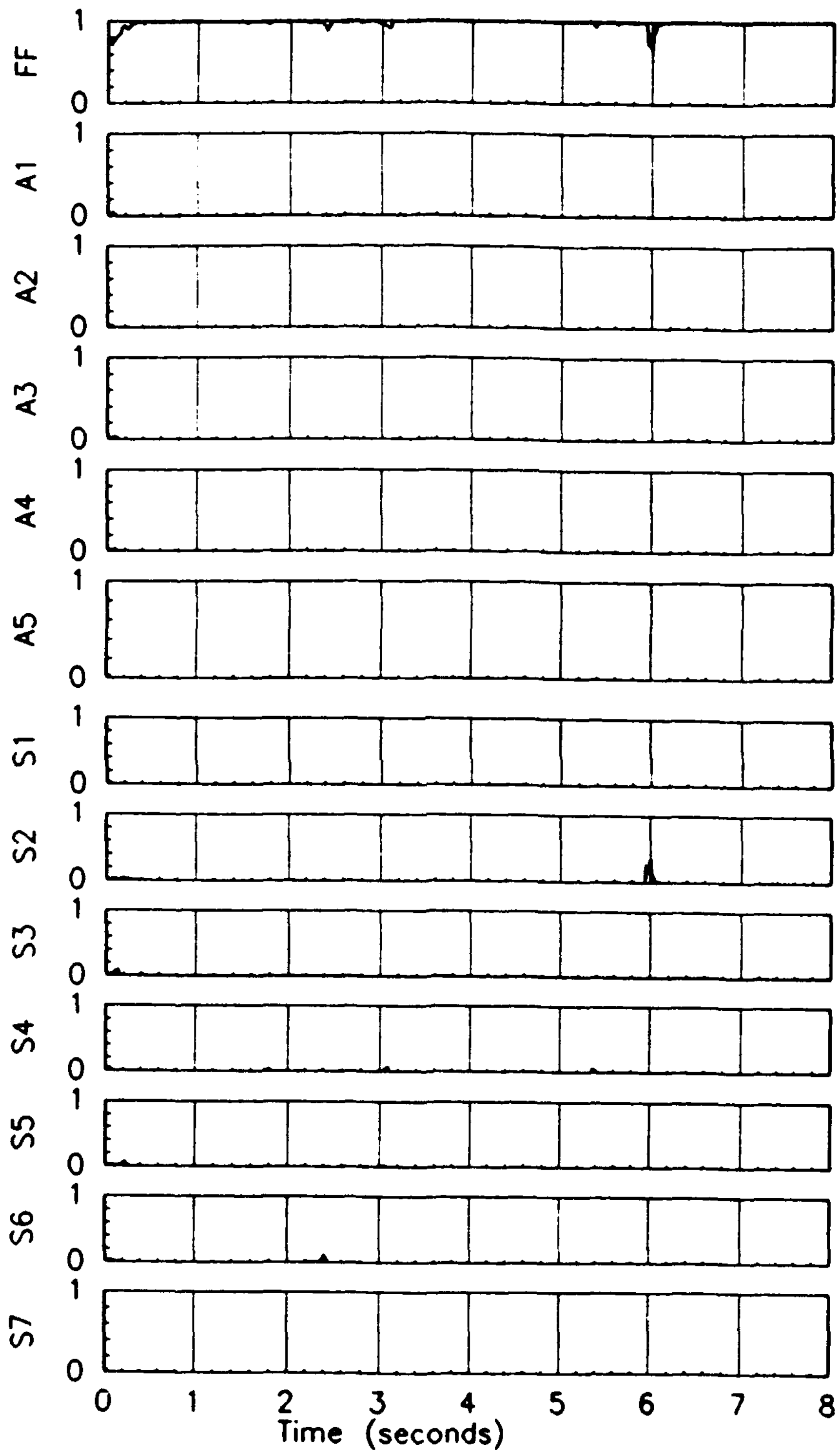
A.28 Probabilities for a yaw rate sensor failure using a purposeful roll and pull command



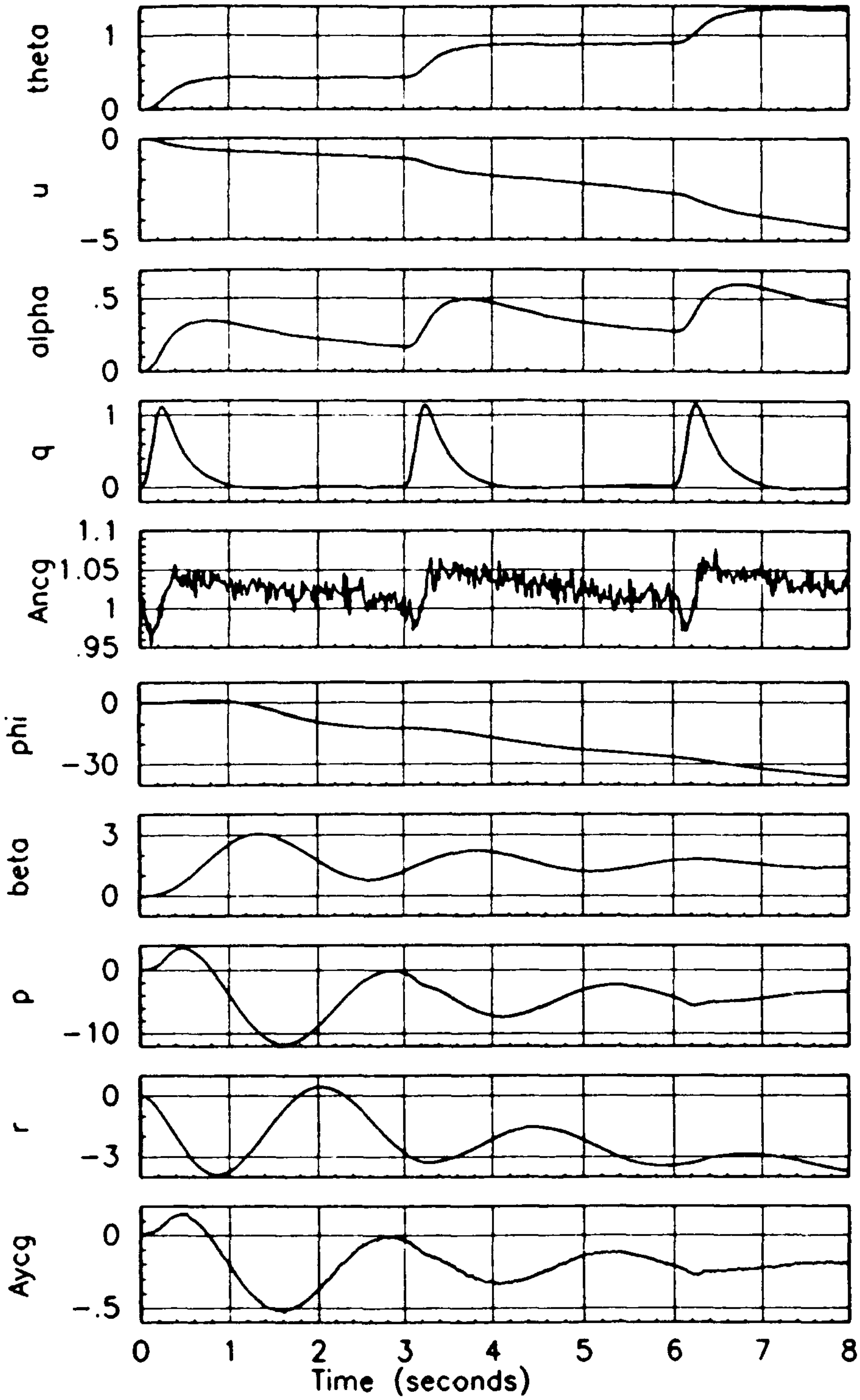
A.29 Probabilities for a lateral acceleration sensor failure using a purposeful roll and pull command

```
      SUBROUTINE COMMAND(FBC,FAC,FPC,T)
C ---- Provides the vista flight control system [CNTRL.FOR]
C       with the command signals for the longitudinal axis
C       [FBC], the lateral command [FAC], and the directional
C       command [FPC].  Additionally, simulation results have
C       indicated the need for a dither signal to "shake up"
C       the system and aid the filters in their identification
C       tasks.

      INCLUDE 'DECLARR.TXT'
      REAL FBC,FAC,FPC,DON,T,omega1,omega2,omega3


      SAVE

C     COMMAND SIGNALS


C . . . . . . . . . . . . . . . . . . . . . . . .
C  Dither signal generation
C    --NOTE:  if a control command is to be used
C             also, it must be added to the below
C             created dither signal.
C . . . . . . . . . . . . . . . . . . . . . . . .

C . . . . . . . . . . . . . . . . . . . . . . . .
       Don=amod(t,3.0)
        IF((Don.ge.0.0).and.(Don.lt.0.125))THEN
          FBC = 14.0
          FAC = -7.0
          FPC = 52.0
        ELSE IF((Don.ge.0.125).and.(Don.lt.0.25))THEN
          FBC =-14.0
          FAC = 8.0
          FPC =-52.0
        ELSE
          FBC = 0.0
          FAC = 0.0
          FPC = 0.0
        END IF

C . . . . . . . . . . . . . . . . . . . . . . . . .
C

c
c     sinusoidal input
c
        omega1 = 15.0
        omega2 = 15.0
        omega3 = 15.0
c
c           FBC= 12.0*sin(omega1*t)
c           IF(FBC.LT.0.0)FBC=12.5*sin(omega1*t)
c           FAC= -11.0*sin(omega2*t)
c           FPC= 30.0*sin(omega3*t)


C
C      IF((Don.ge.0.0).and.(Don.lt.0.1))THEN
C        FBC = 15.2
C        FAC = 16.0
C        FPC = 55.0
C      ELSE IF((Don.ge.0.1).and.(Don.lt.0.125))THEN
C        FBC = (-15.2/0.025)*(Don - 0.1) + 15.2
C        FAC = (-16.0/0.025)*(Don - 0.1) + 16.0
C        FPC = (-55.0/0.025)*(Don - 0.1) + 55.0
C      ELSE IF((Don.ge.0.125).and.(Don.lt.0.3))THEN
C        FBC =-16.5
C        FAC =-14.0
C        FPC =-50.0
C      ELSE IF((Don.ge.0.3).and.(Don.lt.0.325))THEN
C        FBC = (16.5/0.025)*(Don - 0.3) - 16.5
C        FAC = (14.0/0.025)*(Don - 0.3) - 14.0
C        FPC = (50.0/0.025)*(Don - 0.3) - 50.0
```

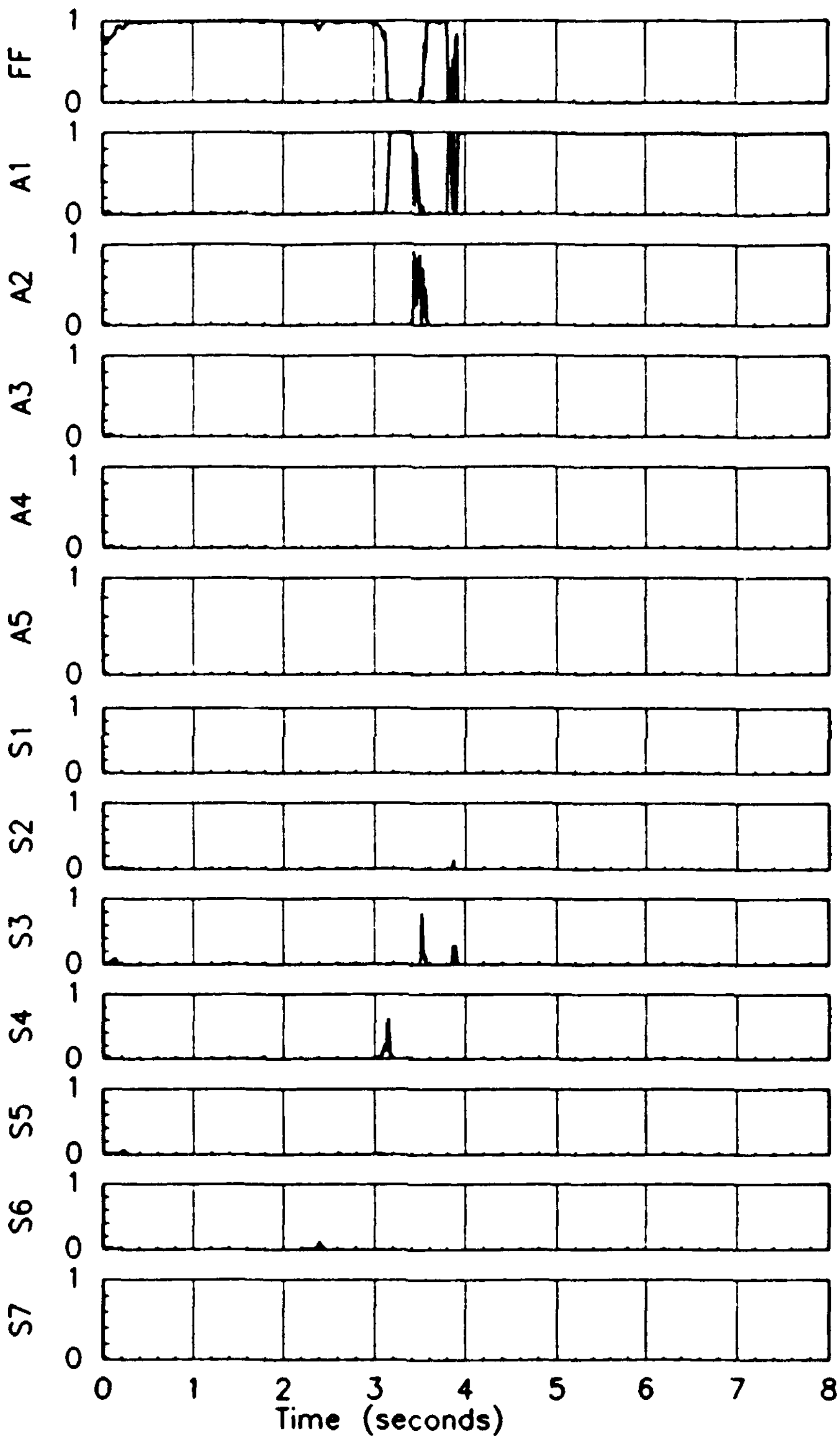
```
c        ELSE
c          FBC = 0.0
c          FAC = 0.0
c          FPC = 0.0
c        END IF

c       Pitch Path

c          IF((t.GT.2.95).and.(t.LT.3.45))THEN
c            FAC=5.0
c          FAC=20.0
           FPC=-40.0
c          ENDIF

c       Roll Path


c       Yaw Path


        RETURN
        END
```



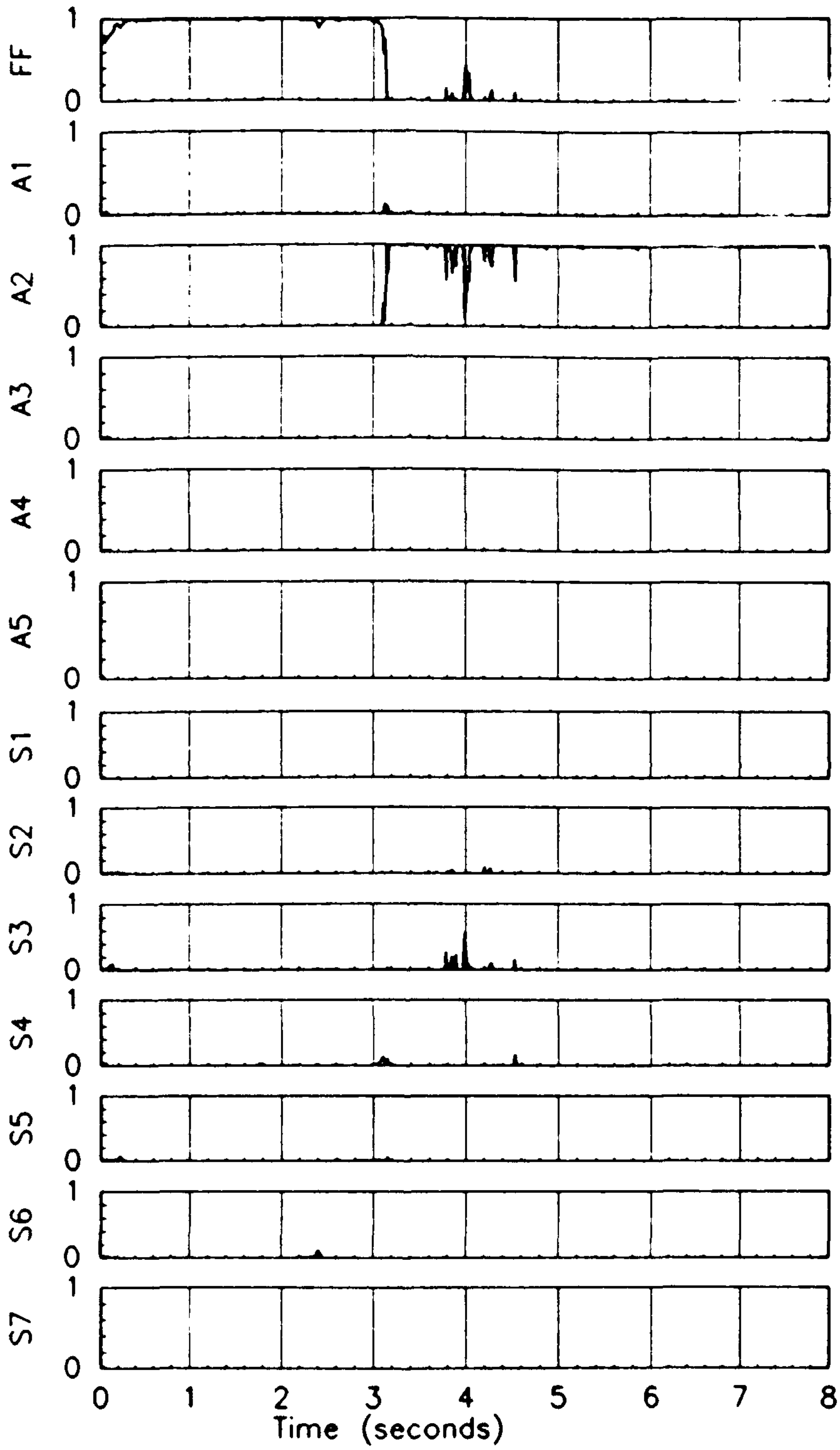
A.30 Probabilities for a no-failure scenario using a purposeful rudder kick and hold command



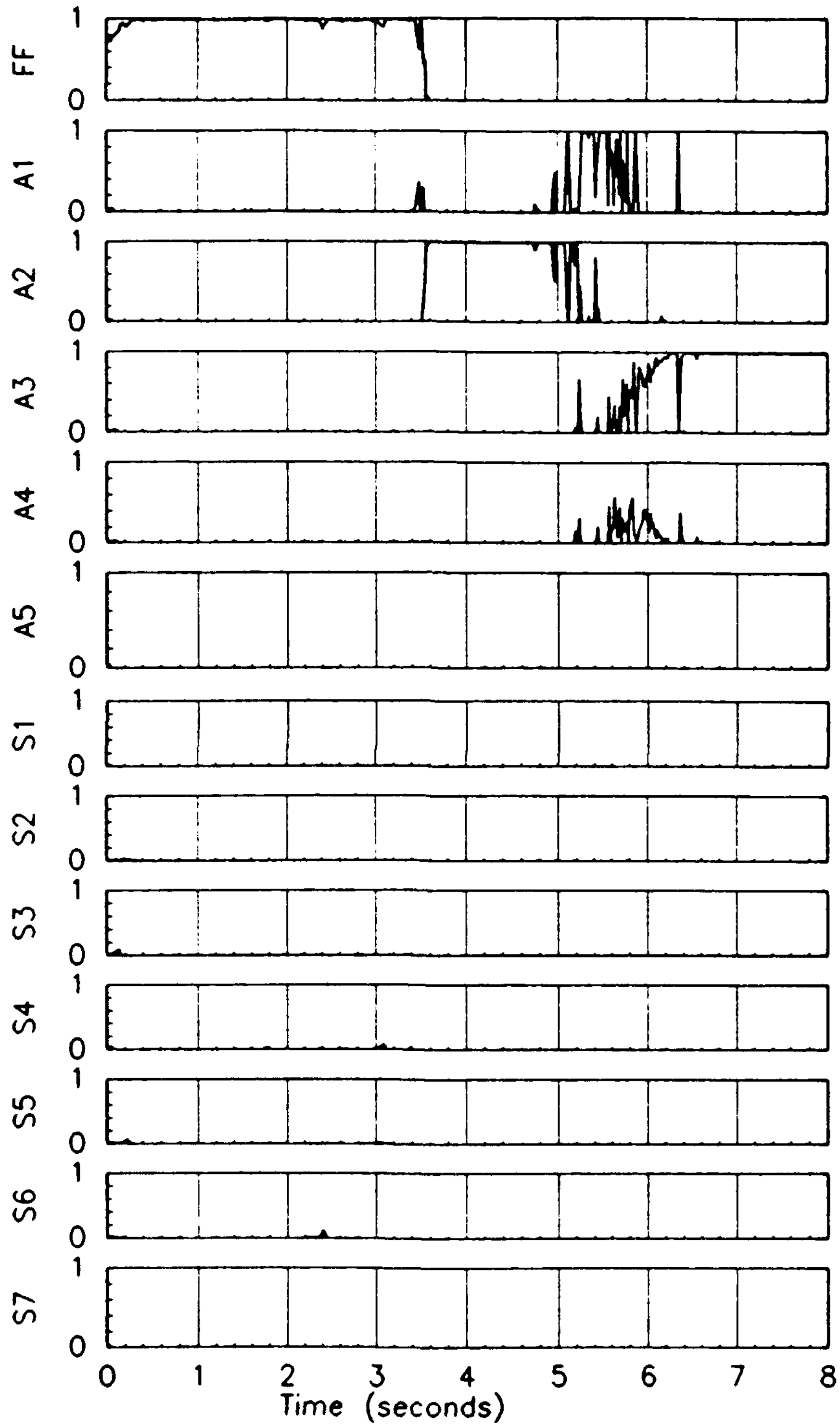
A.31 States for a no-failure scenario using a purposeful rudder kick and hold command



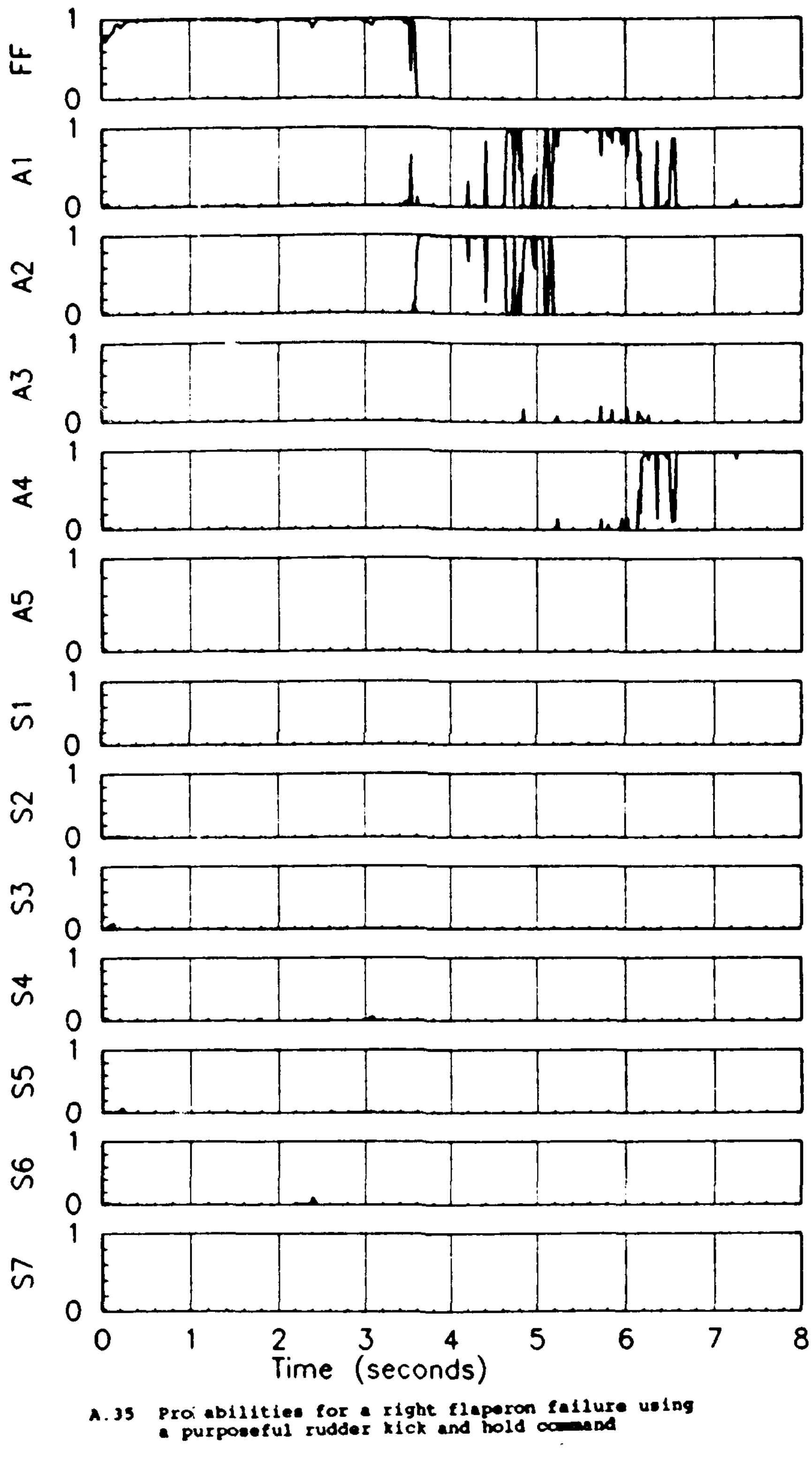
A.32 Probabilities for a left stabilator failure using a purposeful rudder kick and hold command



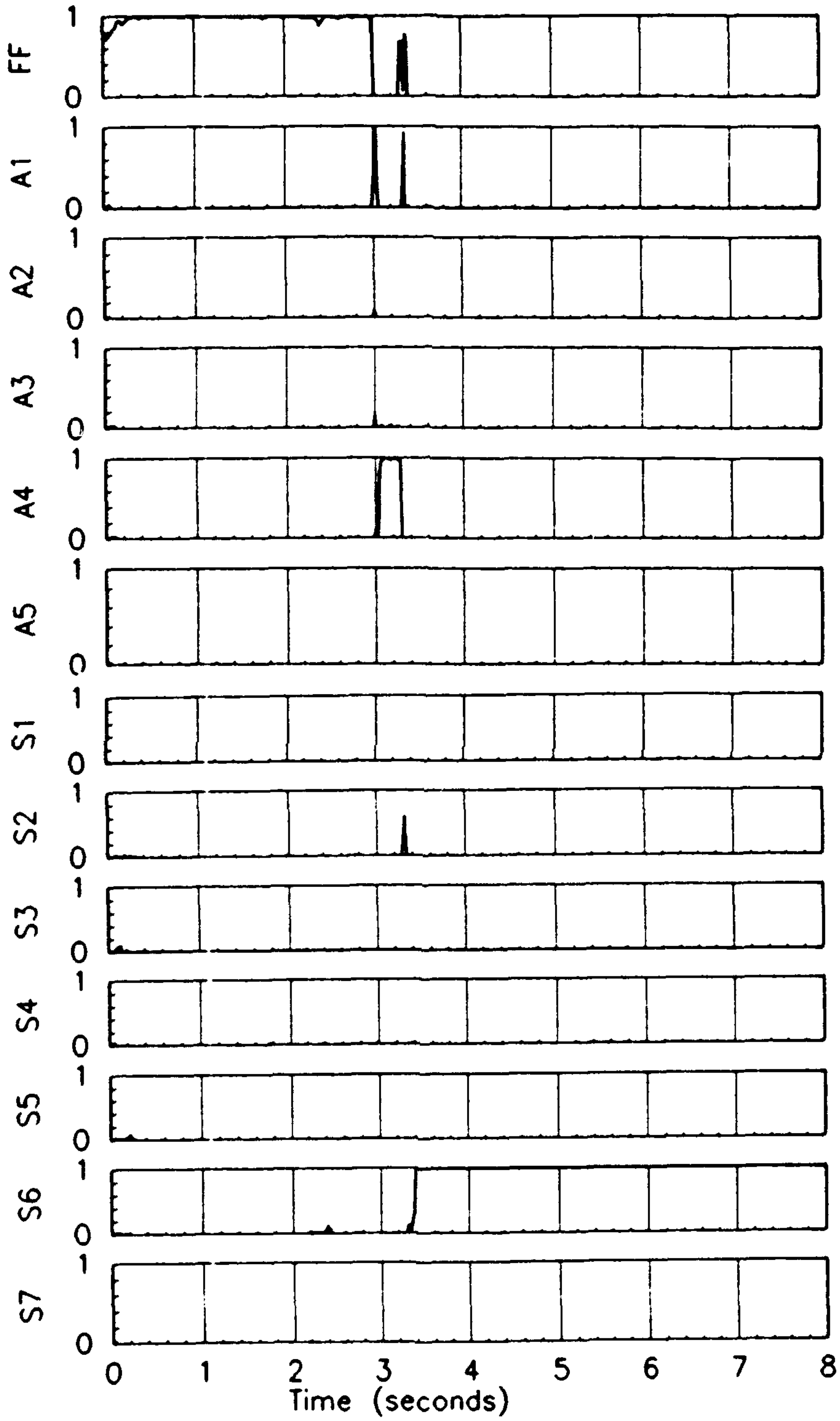
A.33 Probabilities for a right stabilator failure using a purposeful rudder kick and hold command



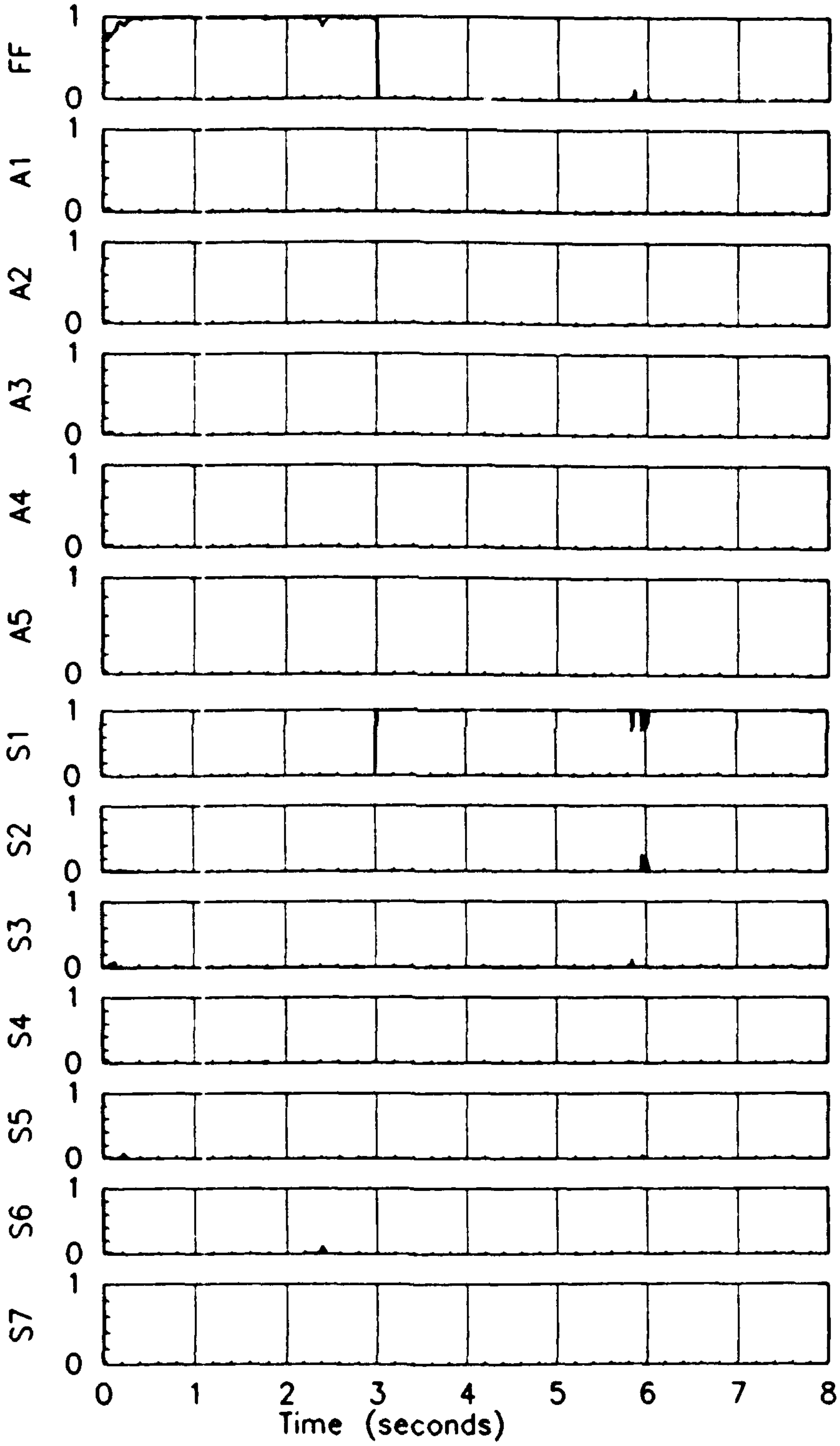
A.34 Probabilities for a left flaperon failure using a purposeful rudder kick and hold command



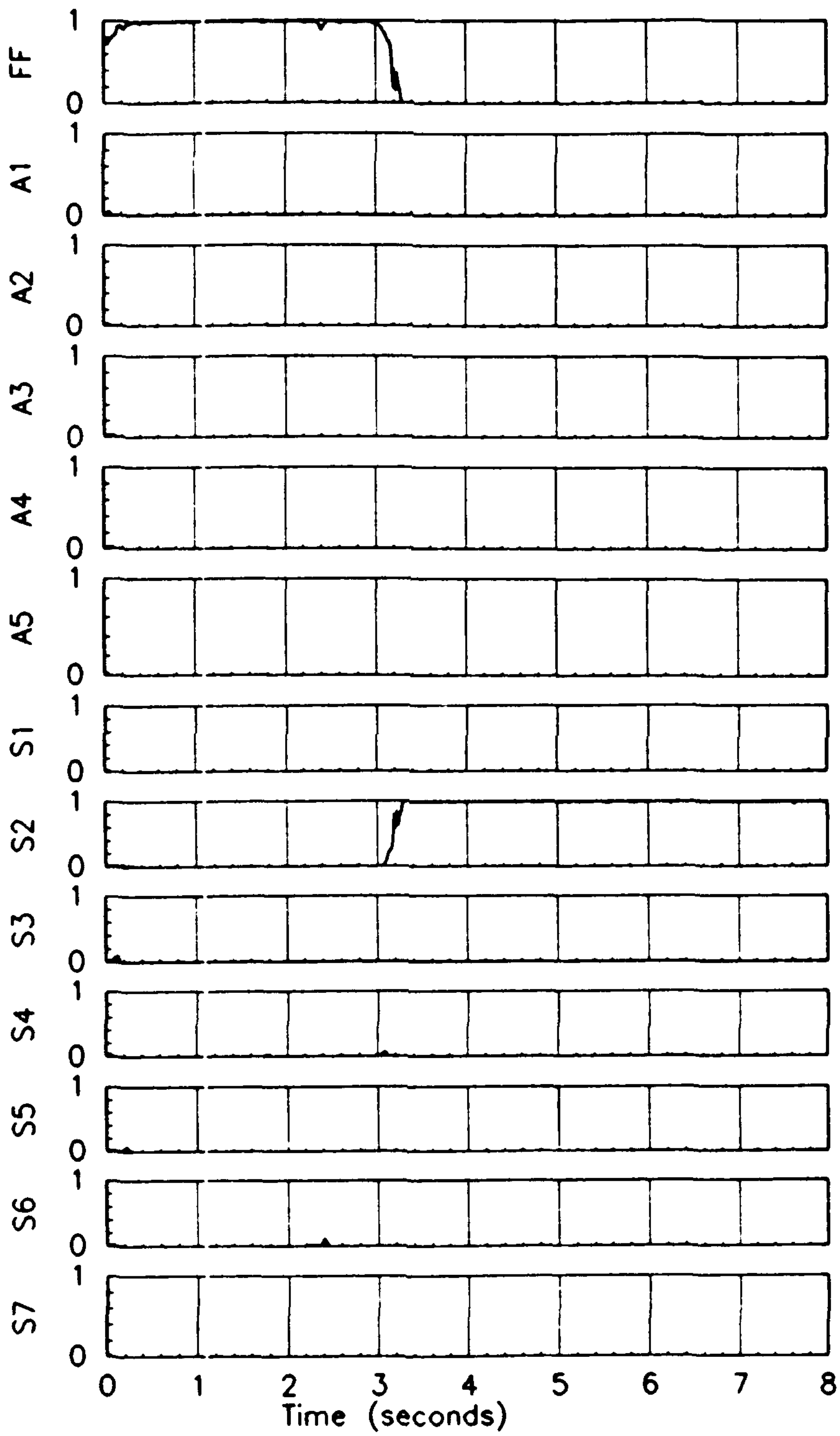
A.35 Probabilities for a right flaperon failure using a purposeful rudder kick and hold command



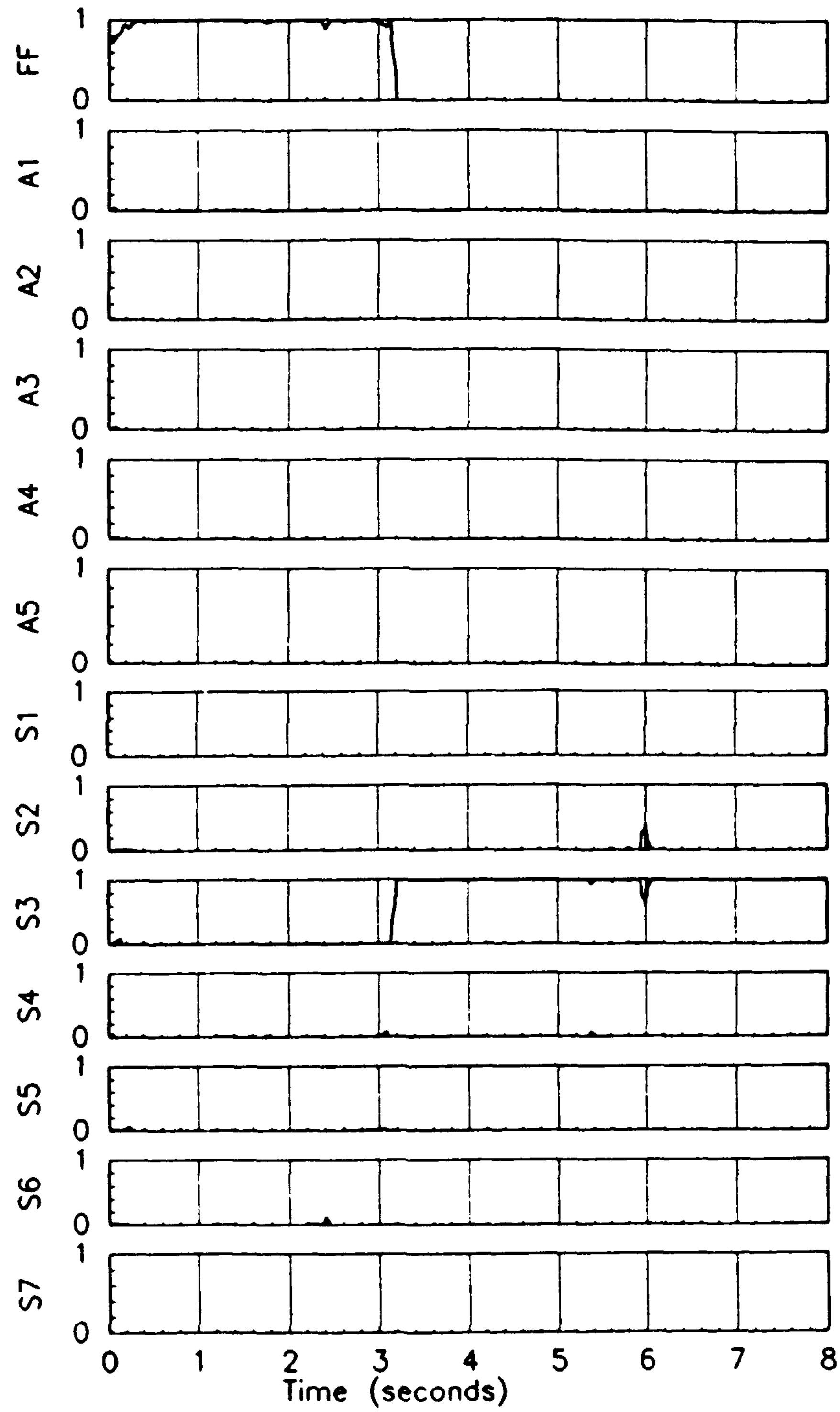
A.36 Probabilities for a rudder failure using a purposeful rudder kick and hold command



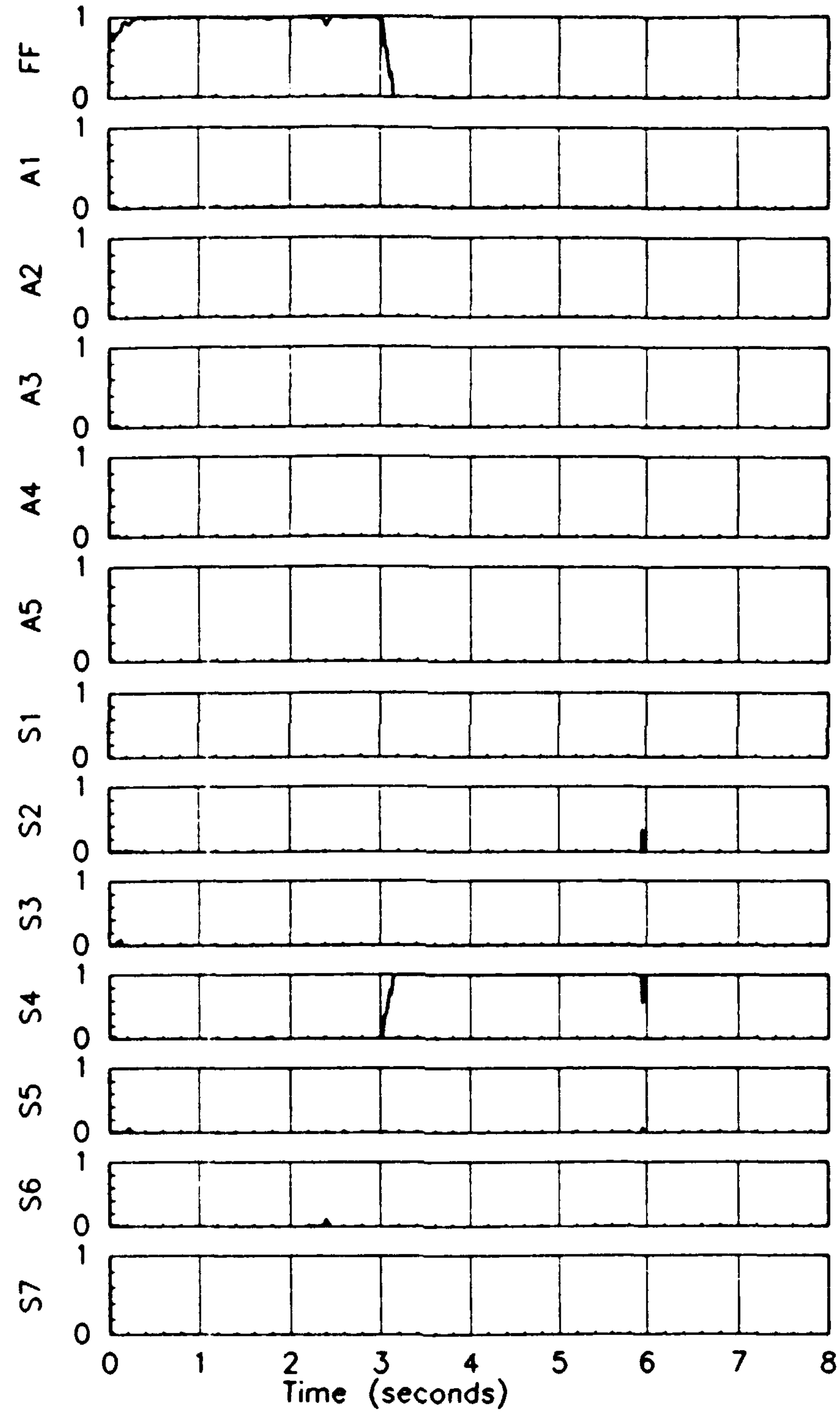
A.37 Probabilities for a velocity sensor failure using a purposeful rudder kick and hold command



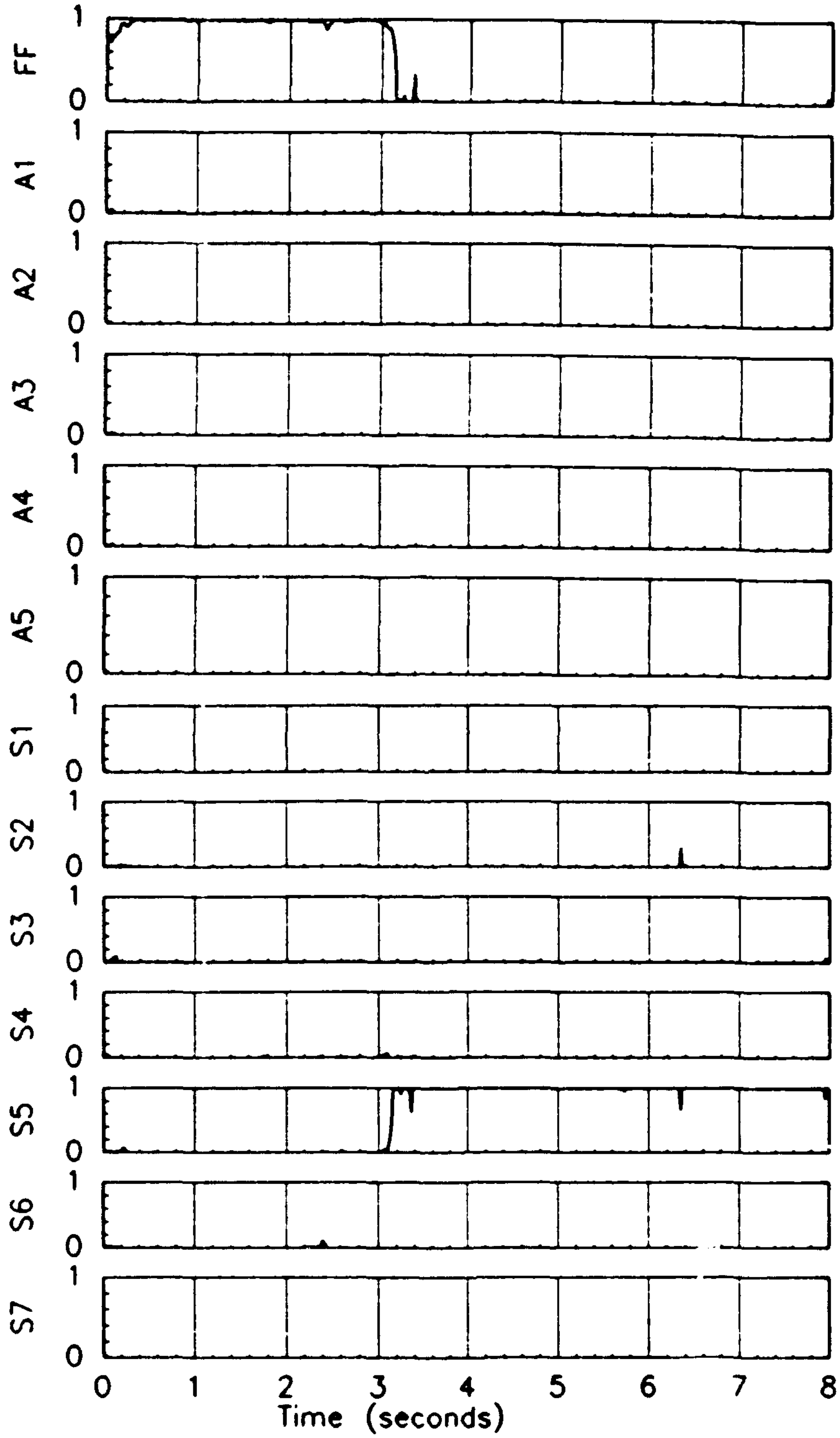
A.38 Probabilities for a angle of attack sensor failure using a purposeful rudder kick and hold command



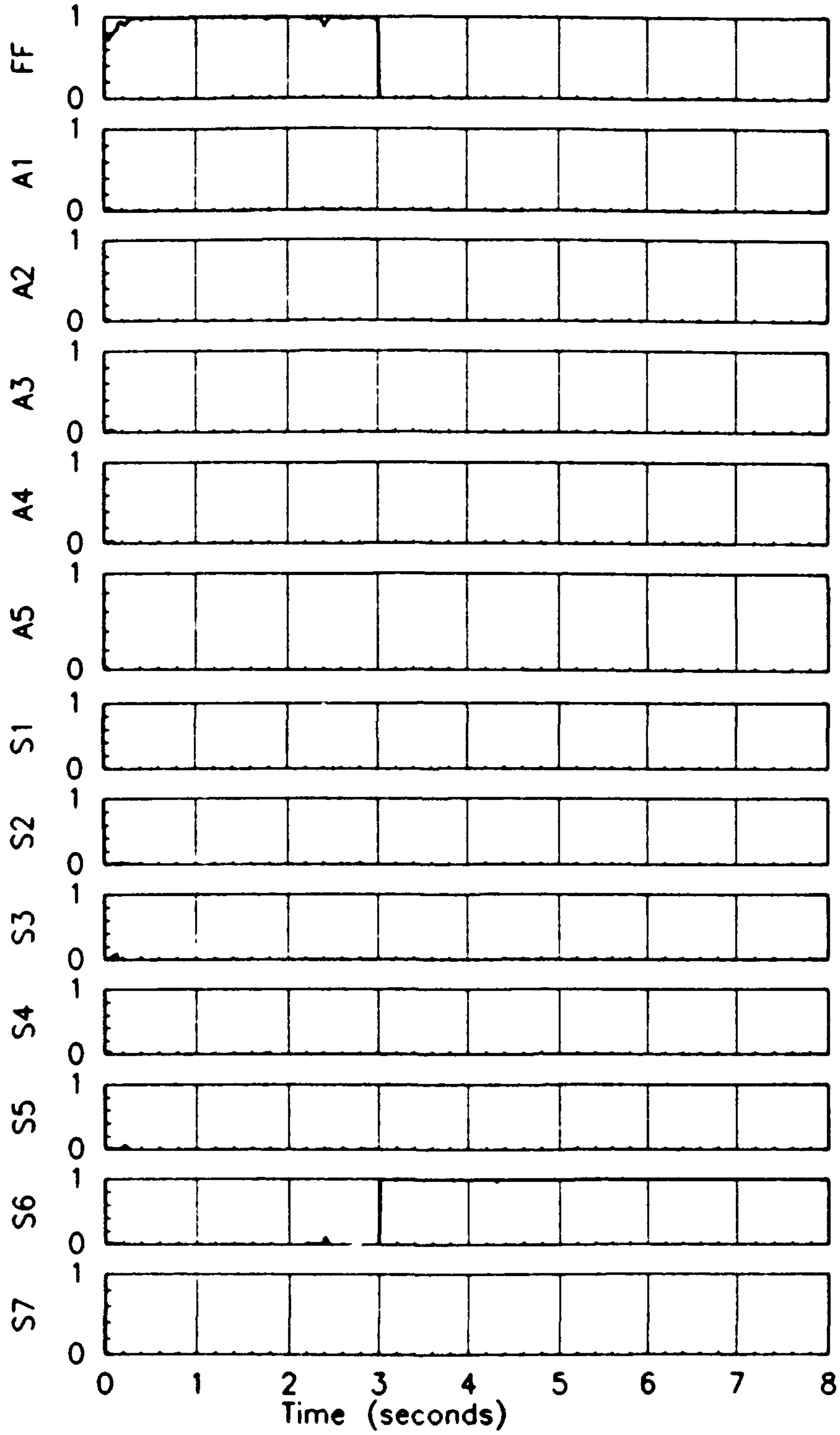
A.39 Probabilities for a pitch rate sensor failure using a purposeful rudder kick and hold command



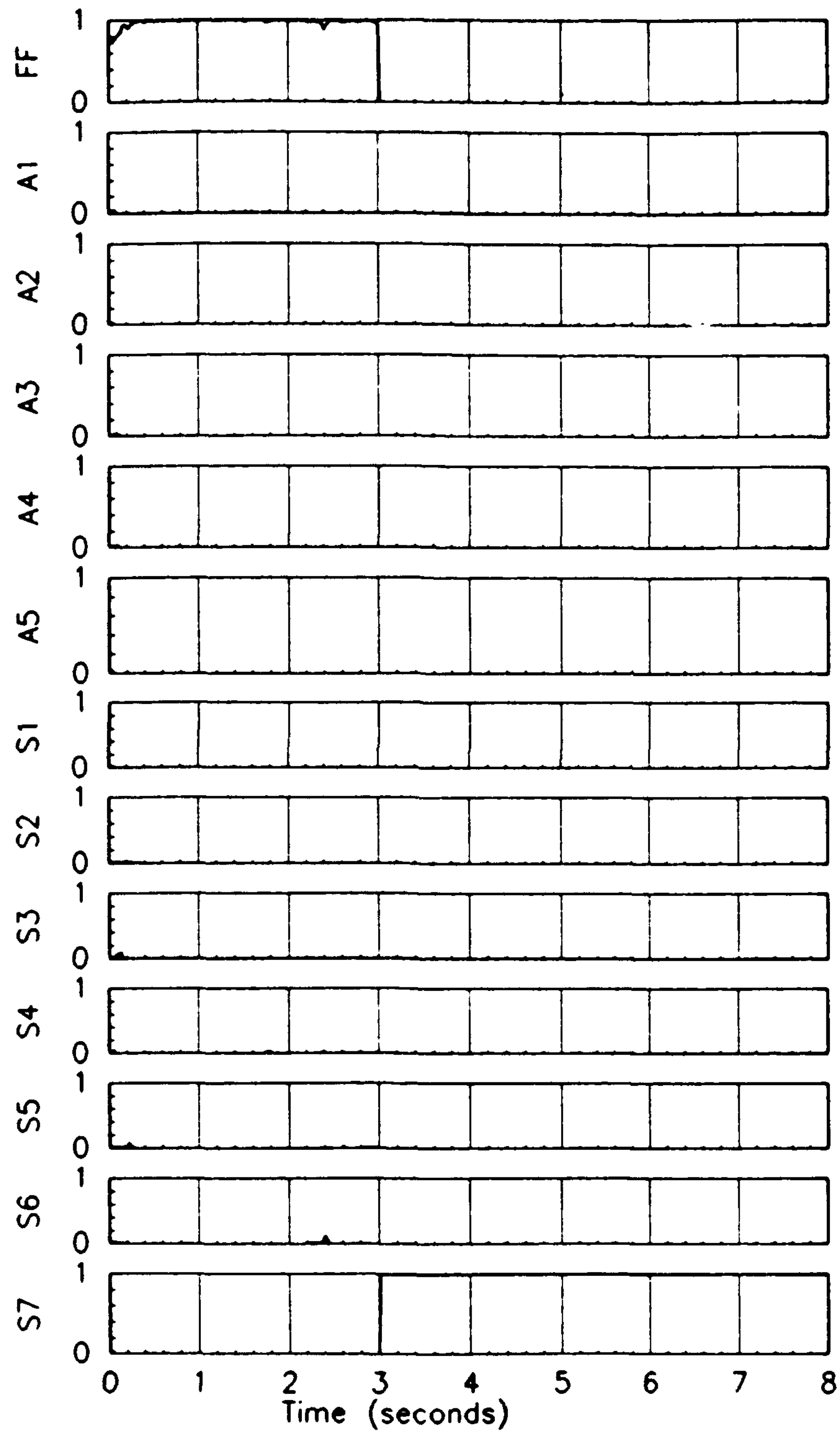
A.40 Probabilities for a normal acceleration sensor failure using a purposeful rudder kick and hold command



A.41 Probabilities for a roll rate sensor failure using a purposeful rudder kick and hold command



A.42 Probabilities for a yaw rate sensor failure using a purposeful rudder kick and hold command



A.43 Probabilities for a lateral acceleration sensor failure using a purposeful rudder kick and hold command

APPENDIX B: VISTA F-16 SIMULATION VERIFICATION RESULTS

The VISTA F-16 simulation developed for this research effort was validated using the GENESIS simulation. The GENESIS simulation resides in the Flight Dynamics Laboratory at Wright-Patterson AFB, OH. The GENESIS simulation is a nonlinear six-degree-of-freedom simulation. The GENESIS code allows the user to to run a simulation using the non-linear aerodynamic model or an aerodynamic model linearlized about a trim point. The code produces laser quality traces of any simulation variables. The code also allows the examination of internal flight control system variables at any time in the simulation. The aerodynamic model used in the thesis effort was a model linearized about the trim point of 0.4 Mach, 20000 ft.

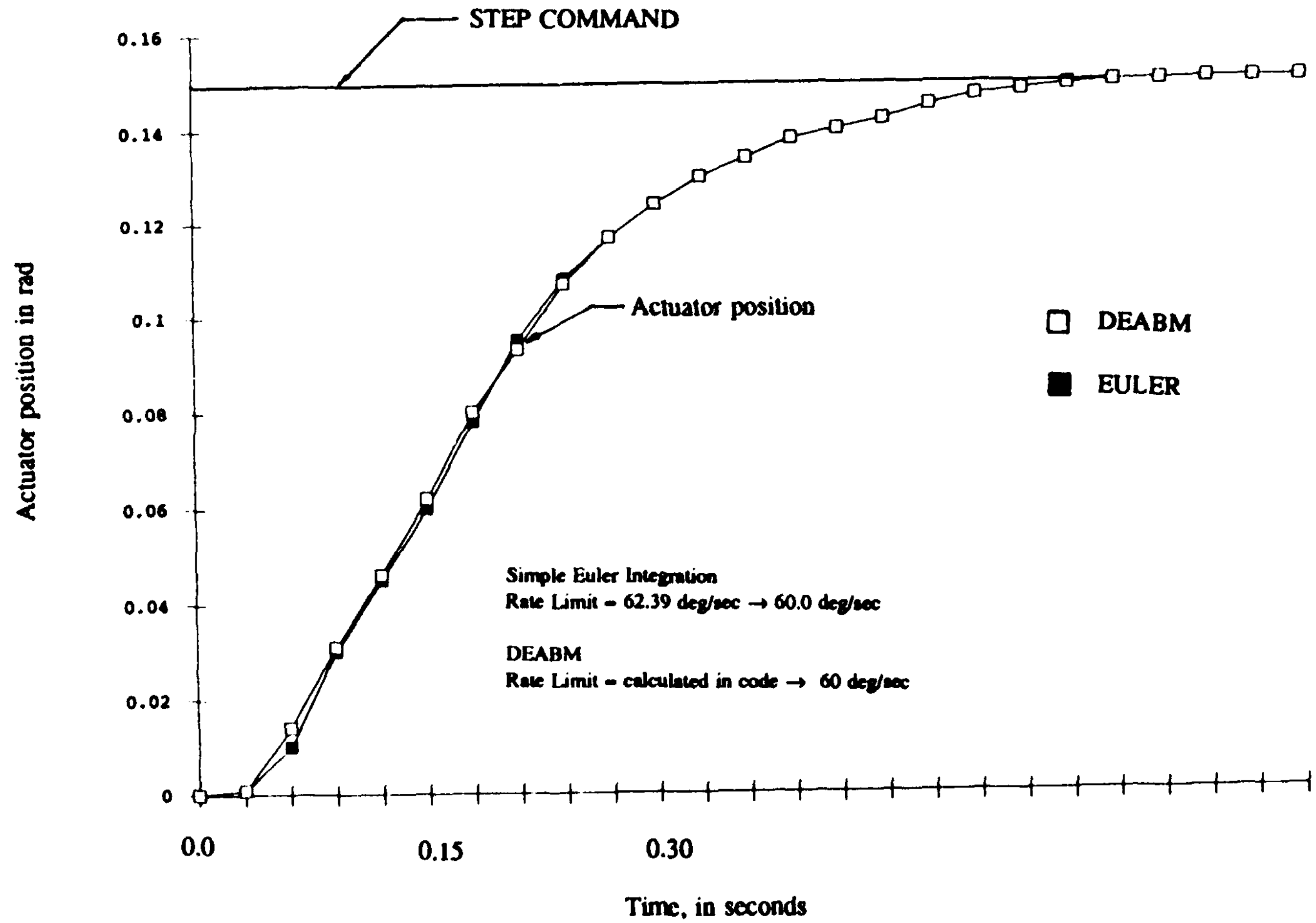
The flight control system was checked by applying the linearized aerodynamic data base from the GENESIS simulation. After the open loop flight control system was checked out, the system was run in the closed loop mode with the linearized aerodynamic data base. These plots are shown in Figures B.1 - B.17. For the 8 second time plots, the correlation is very good. Differences in the simulation are due to the comparison of a linear and nonlinear data base and the result of modeling the flight control system without the higher order dynamics (above 40 rad/sec). Actuator models were checked separately by evaluating each of the models with theoretical predictions and varying sample rates. The integration routine DEABM was evaluated with a comparison of results from a fast-sampling simple Euler integration program. The test were conducted on the 4th order acutator models. The results verifed the integration routines and demonstrated that the rate limiting functions were correctly modeled within the EOM subroutine (Figure B.1). Figure B.2 demonstrates the pitch command used to evaluate the coded flight control system at the 0.8 Mach, 10000 ft test case (high dynamic pressure case tested by Stratton [13]. The command is a simple pitch pull and hold of 10 lbs for a duration of 2 seconds. Figure B.3 demonstrates the results of the simulation as compared to the GENESIS nonlinear simulation. The CNTRL subroutine, developed for this thesis, is the line which contains plotted data points. From Figure B.3, the Mach or velocity data demonstrates a marginal match. Figure B.4 presents the normal acceleration trace as a function of time. Comparision of the two plots yields small mismatches at 2.0 and 3.0 seconds. Figure B.5 presents the angle of attack as a function of time. Comparision of the data demonstrates

excellent results. Figure B.6 displays the pitch angle with time. Again, comparision of the data demonstrates excellent results. Figure B.7 demonstrates the pitch rate as a function of time. Very small mismatches occur near the relative minima and maxima of the traces. Overall the correlation is very good. Figures B.2 through B.7 presented data for a 10 lb pull and hold for a duration of 2.0 seconds. The 10 lb pull was chosen to exercise the simulation without violating the small angle criteria and to maintain linear assumption validity. The next verification case demonstrates a 29 lb pull and hold at 0.8 Mach and 10000 ft. Figure B.8 demonstrates the Mach number as a function of time. The data for the CNTRL subroutine was not plotted beyond 5 seconds since it is clear that the velocity has diverged far from the GENESIS code. This divergence is attributed to the simple drag model used within the linear model. Figure B.9 demonstrates the Normal Acceleration (in ft/sec^2) vs time. The plots are comparable until about 2 seconds. Figure B.10 displays the angle of attack for this test case. Again, a divergence between the curves occurs at approximately 2.0 seconds. Figure B.11 presents the data for the pitch angle. While the rate of divergence is smaller, divergence is present. Figure B.12 demonstrates the pitch rate as a function of time. The curves are comparable through the first transient (approximately 1.8 seconds). Figures B.8 though B.12 were presented to provide a boundary. Figures B.2 through B.7 provided an operating condition and Figures B.8 through B.12 provided the reader with an expectation of performance degradation for venturing too far from that operating condition. Figure B.13 provides data for a 29 lb longitudinal stick pull and hold for 0.4 Mach at 20000 ft. While the Mach is slightly biased, the curve is reasonably close to the GENESIS data. Figure B.14 demonstrates the altitude as a function of time. The altitude is 20 ft in error after 4.0 seconds. Figure B.15 demonstrates the angle of attack as a function of time. The results are reasonably close up to 3.0 seconds. The general shape of the curve appears correct. Figure B.16 demonstrates the pitch angle with time. The results are within 1 degree after 3 seconds and 4 degrees after 4 seconds. The bounds for the 0.4 Mach at 20000 ft case are considerably better than the 0.8 Mach at 10000 ft case. This is true for the operating conditions as well. Figure B.17 demonstrates the open loop check out of the CNTRL code with the GENESIS code. The nonlinear data base was transferred and input into the CNTRL subroutine as the input data. Figure B.17 demonstrates the effect of not including the higher order dynamic terms.

In this thesis the majority of the identification effort was done with moderate magnitude, small time duration

dither signals. A comparison with the GENESIS code should produce nearly identical results. The purposeful commands are the exception to the rule. A few purposeful commands resulted in pitch angles of over 30 degrees after 8 seconds. The majority of commands are within the small angle constraints and near the trim conditions.

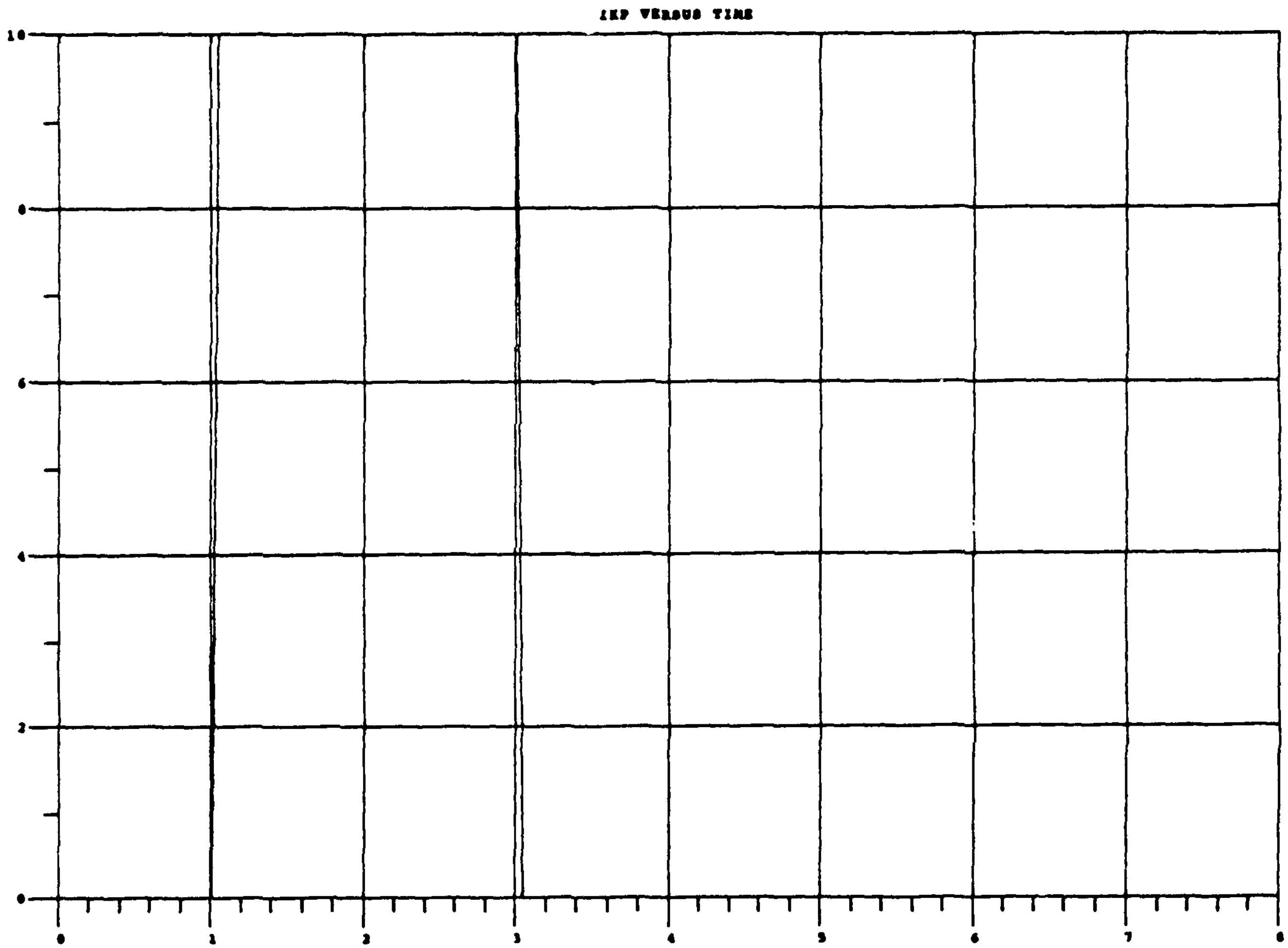
Check of subroutine EOM - Step input of -0.15 rad
DEABM routines vs Simple Euler Integration (0.0015)



B.1 Verification of actuation limiting in subroutine EOM

F-16 MODEL

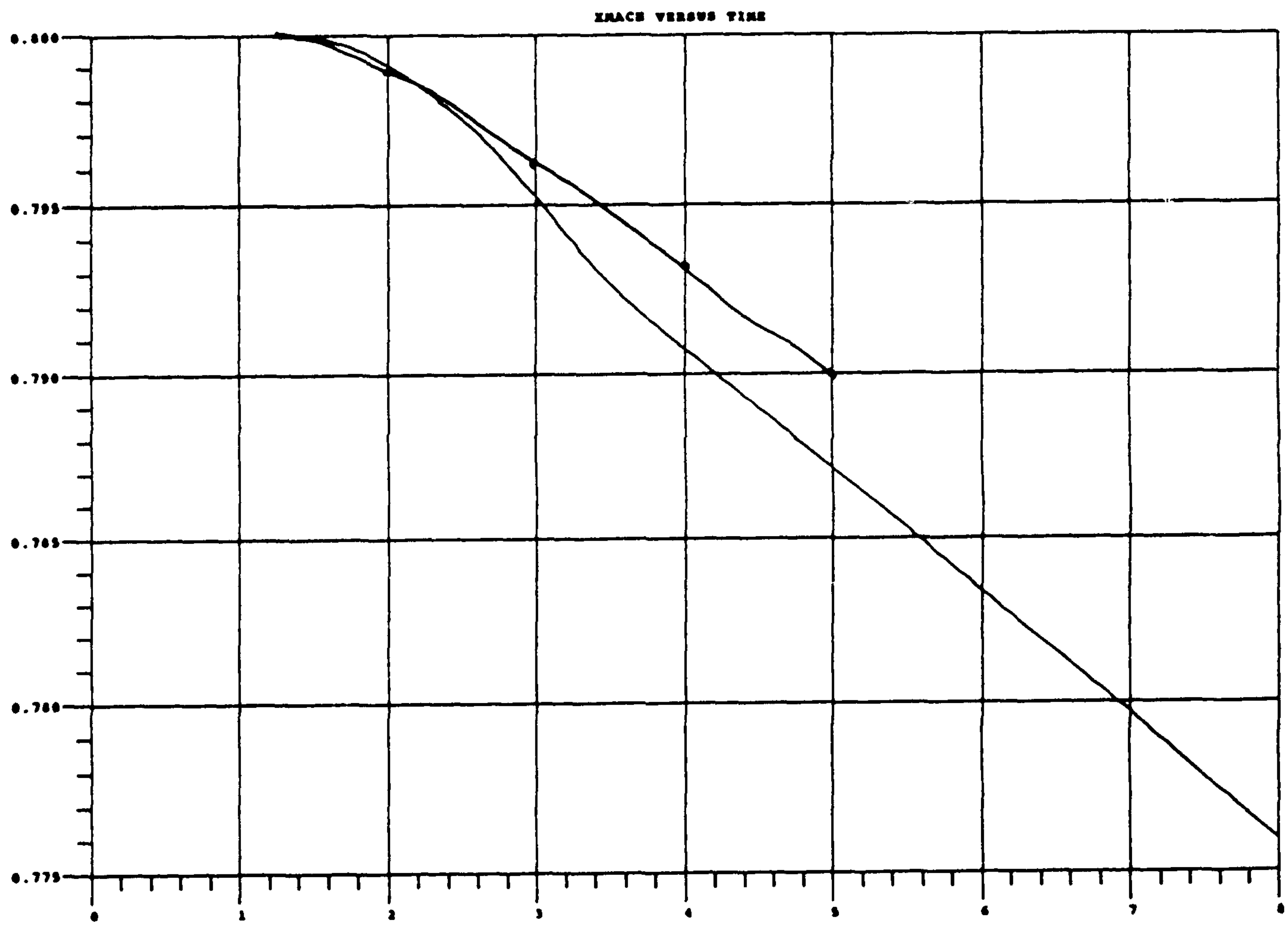
19-AUG-91



B.2 Longitudinal stick input for 0.8 Mach 10000 ft check case

F-16 MODEL

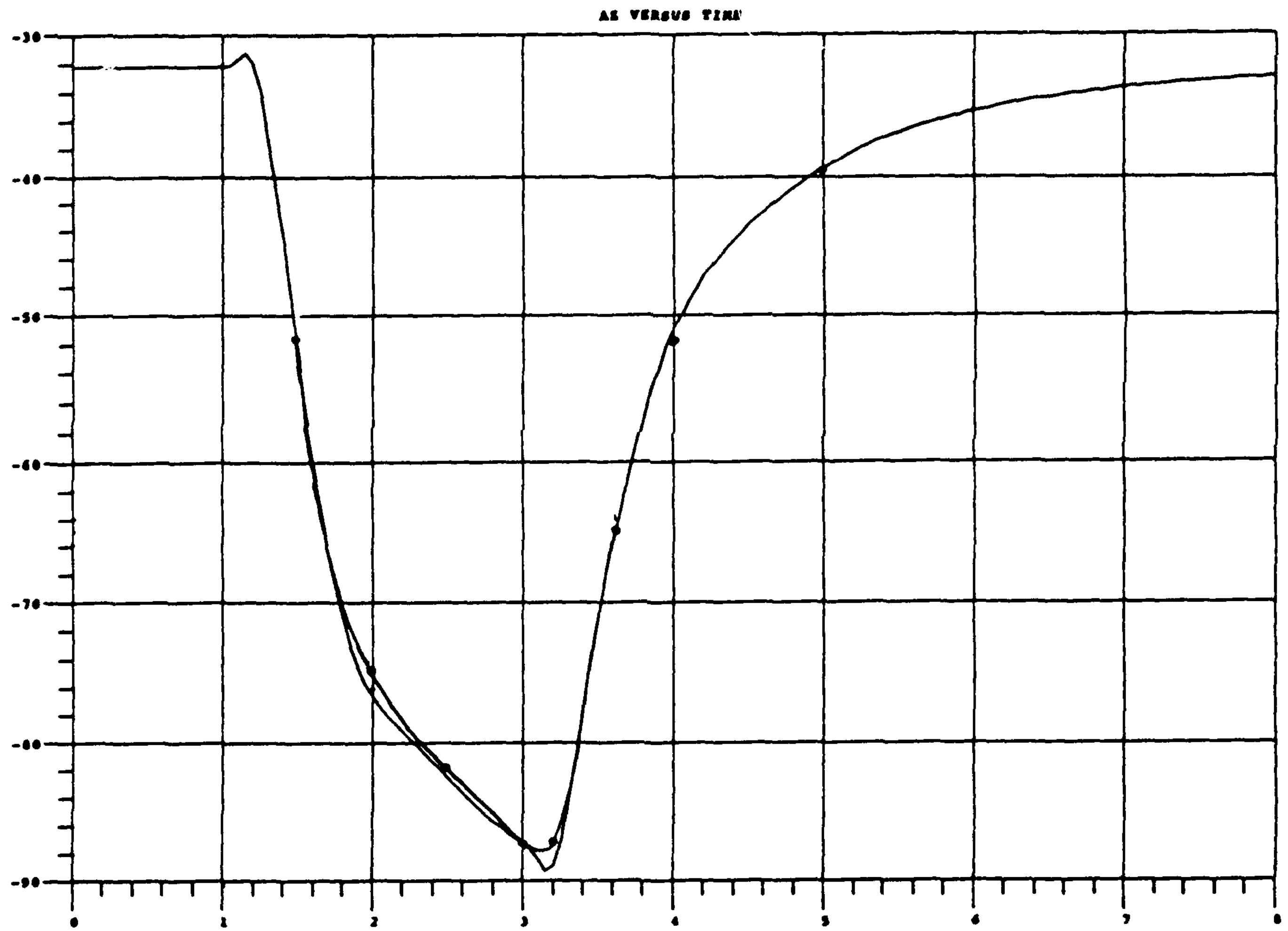
19-AUG-91



B.3 Mach vs time for 10 lb longitudinal stick pull of duration 2 seconds for 0.8 Mach 10000 ft check case

F-16 MODEL

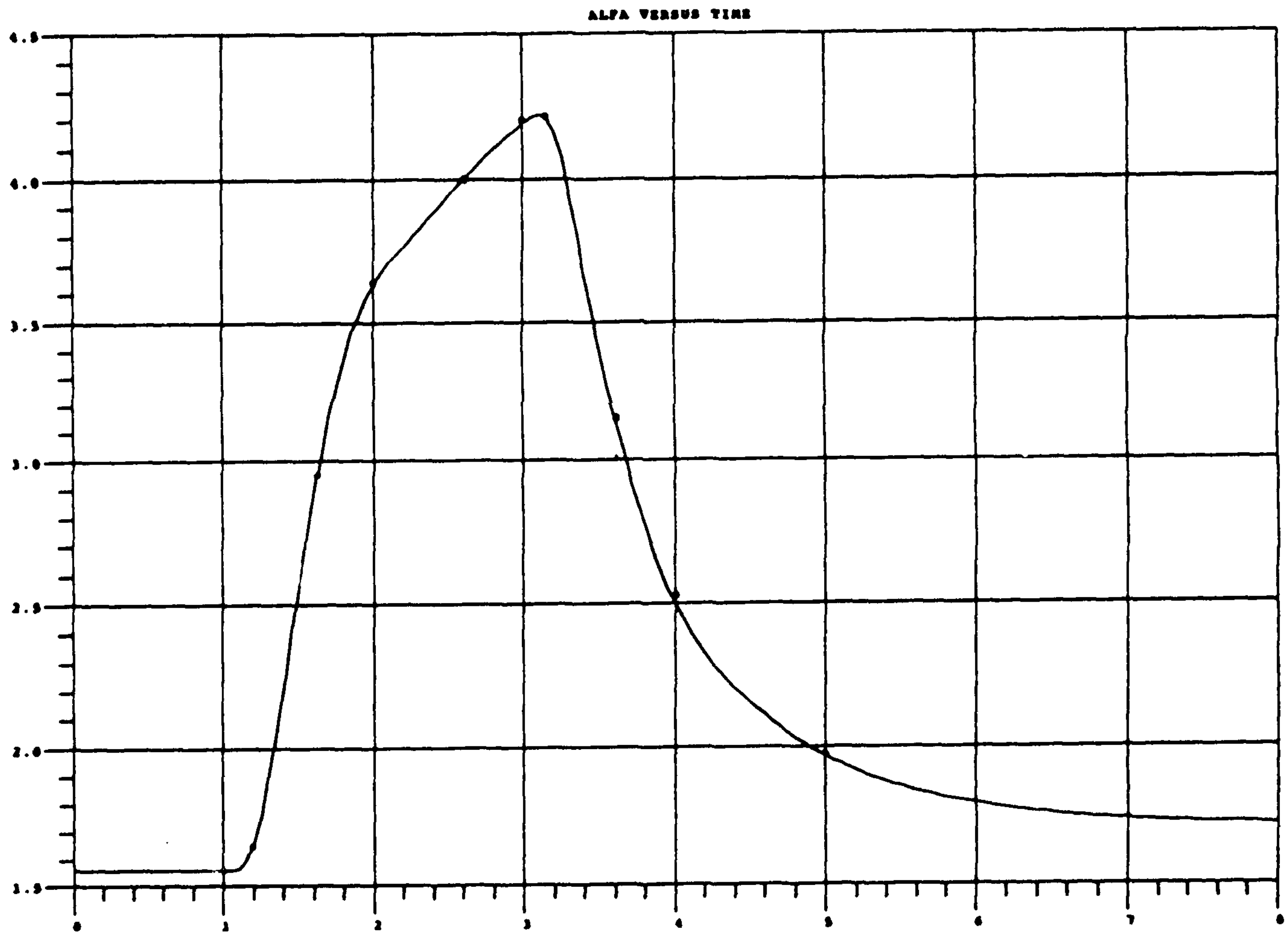
19-AUG-91



B.4 Normal acceleration vs time for 10 lb longitudinal stick pull of duration 2 seconds for 0.8 Mach 10000 ft check case

F-16 MODEL

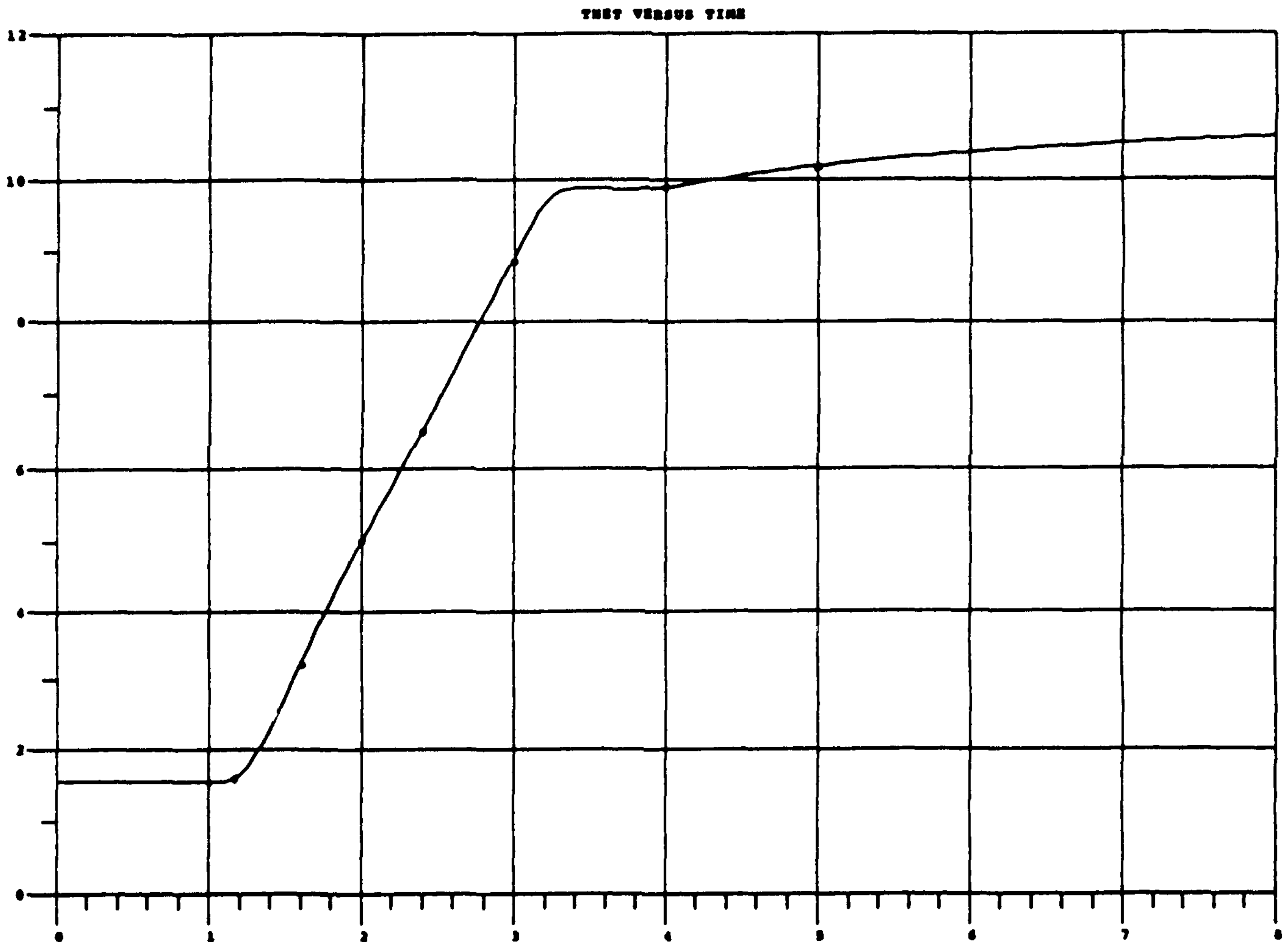
19-AUG-91



B.5 Angle of attack vs time for 10 lb longitudinal stick pull of duration 2 seconds for 0.8 Mach 10000 ft check case

F-16 MODEL

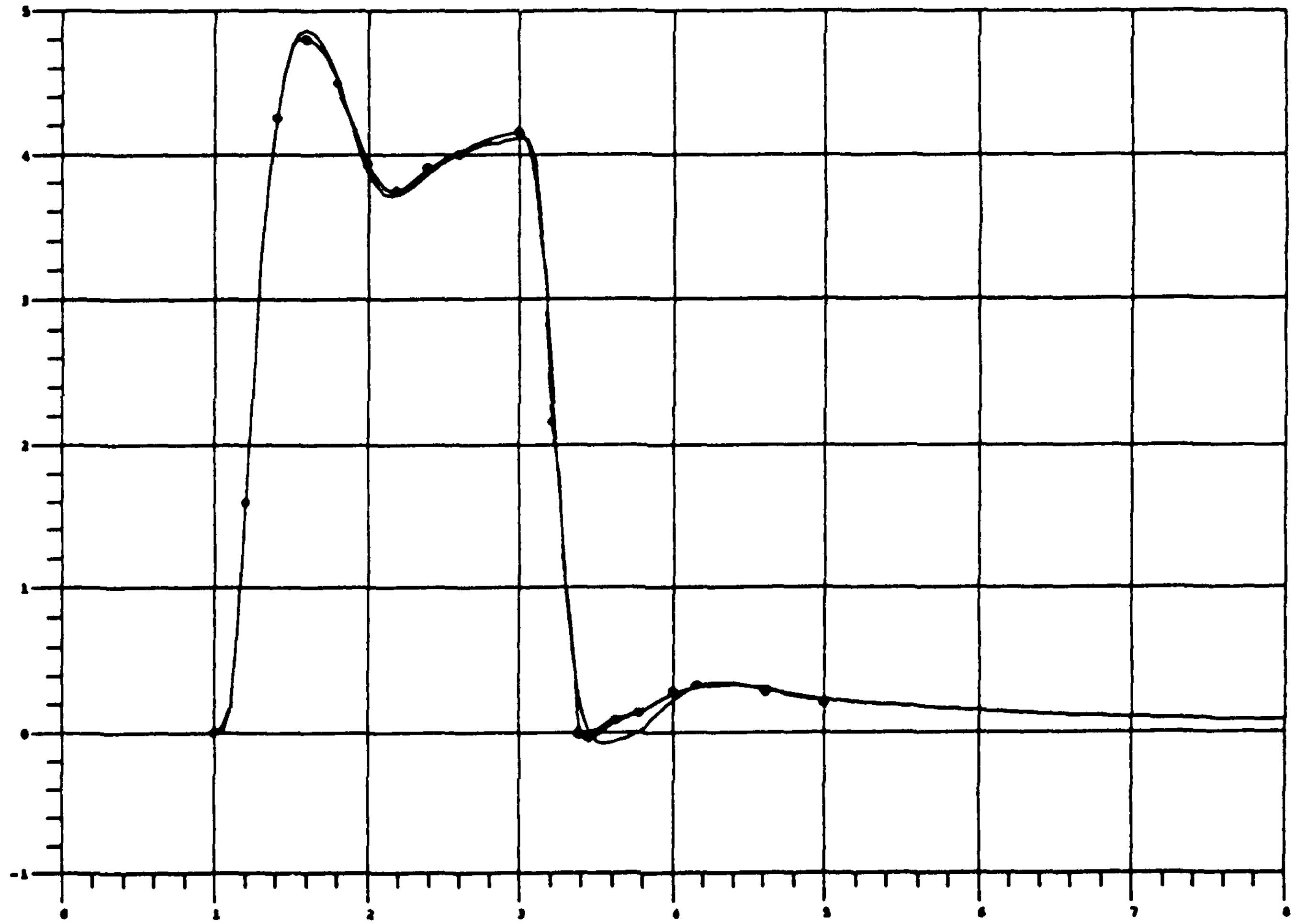
19-AUG-91



B.6 Pitch angle vs time for 10 lb longitudinal stick pull of duration 2 seconds for 0.8 Mach 10000 ft check case

F-16 MODEL

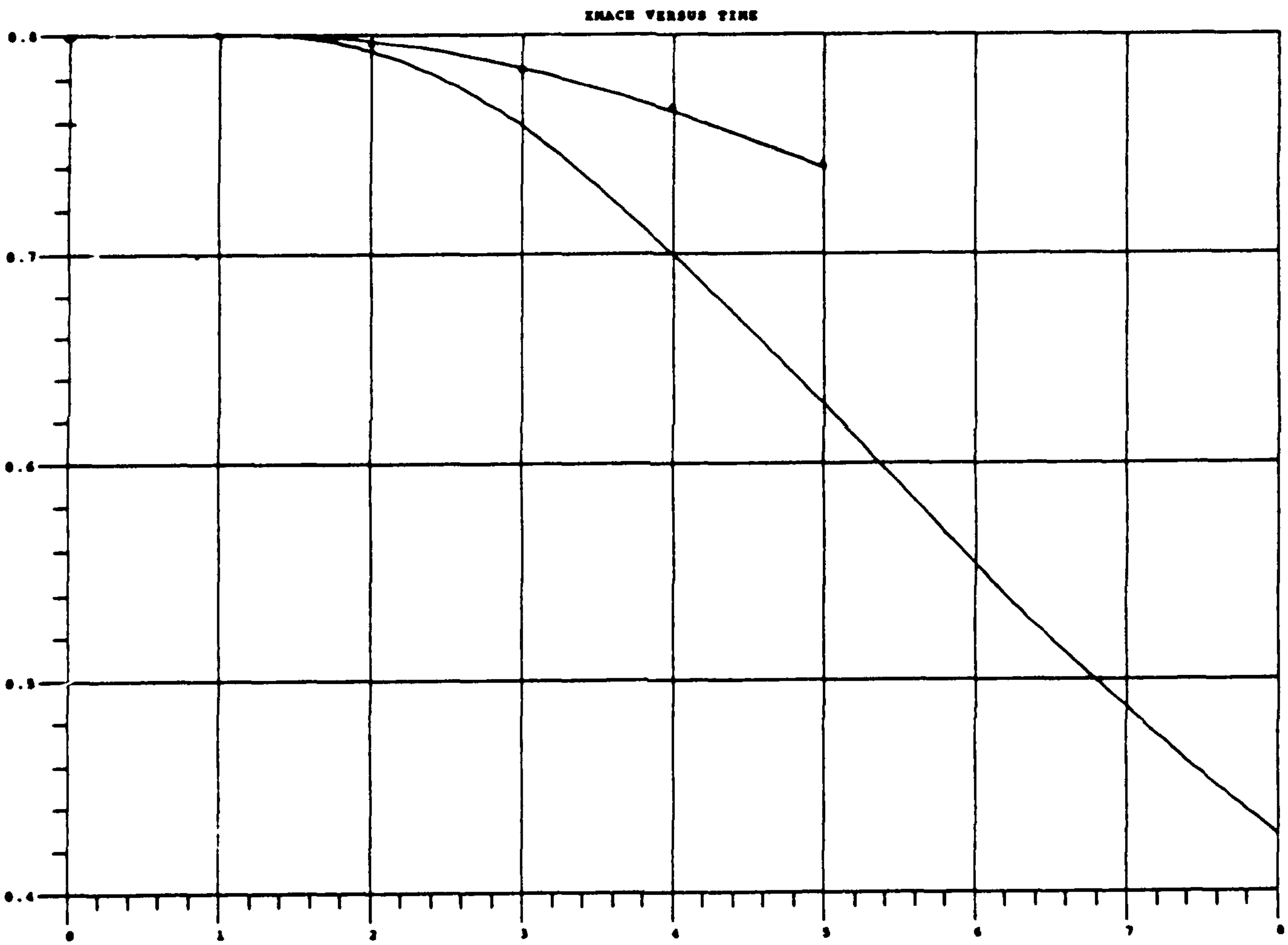
19-AUG-91



B.7 Pitch rate vs time for 10 lb longitudianl stick pull of duration 2 seconds for 0.8 Mach 10000 ft check case

F-16 MODEL

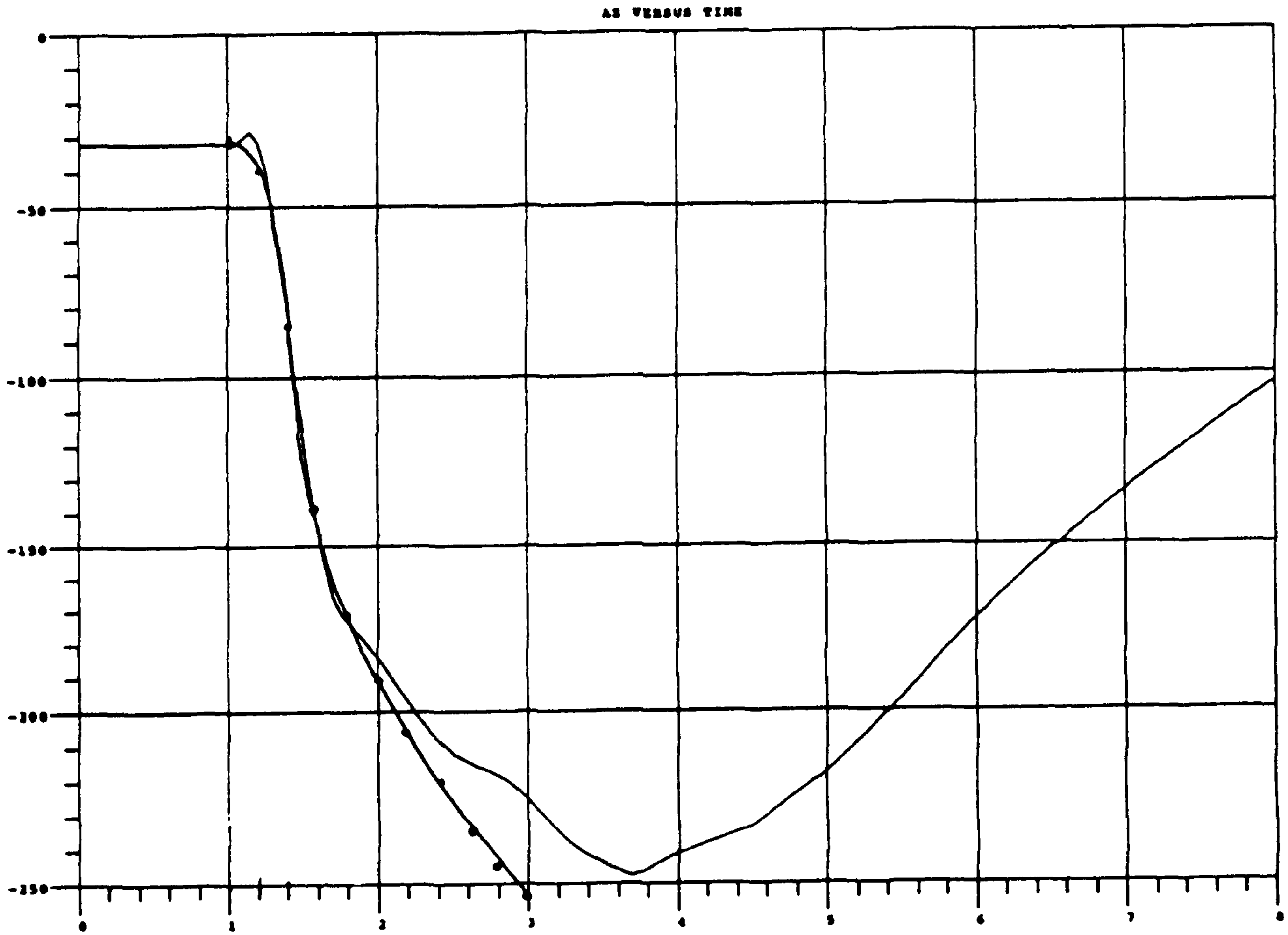
23-APR-91



B.8 Mach vs time for 29 lb longitudinal stick pull and hold for 0.8 Mach 10000 ft check case

F-16 MODEL

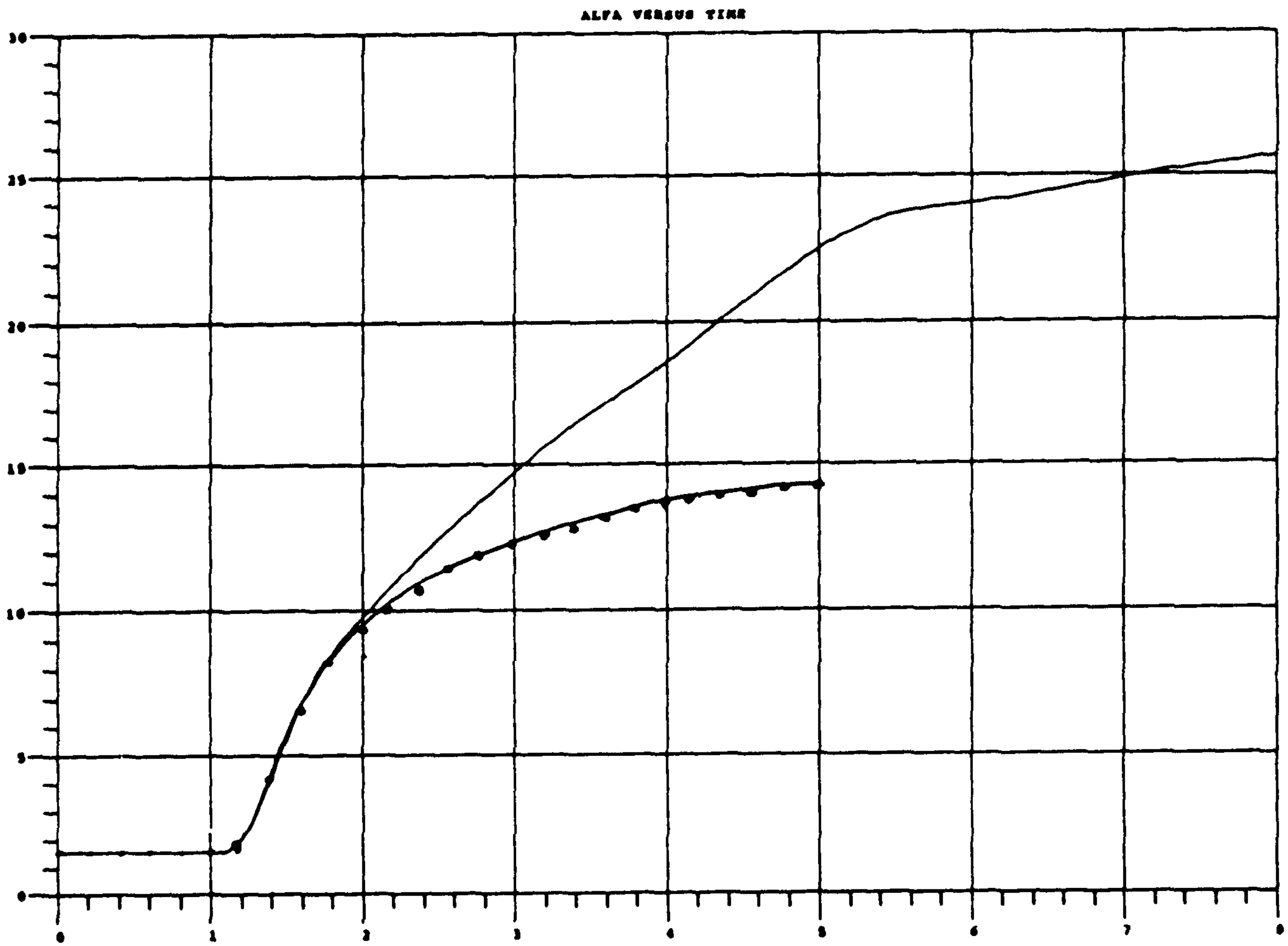
23-APR-91



B.9 Normal acceleration vs time for 29 lb longitudinal stick pull and hold for 0.8 Mach 10000 ft check case

F-16 MODEL

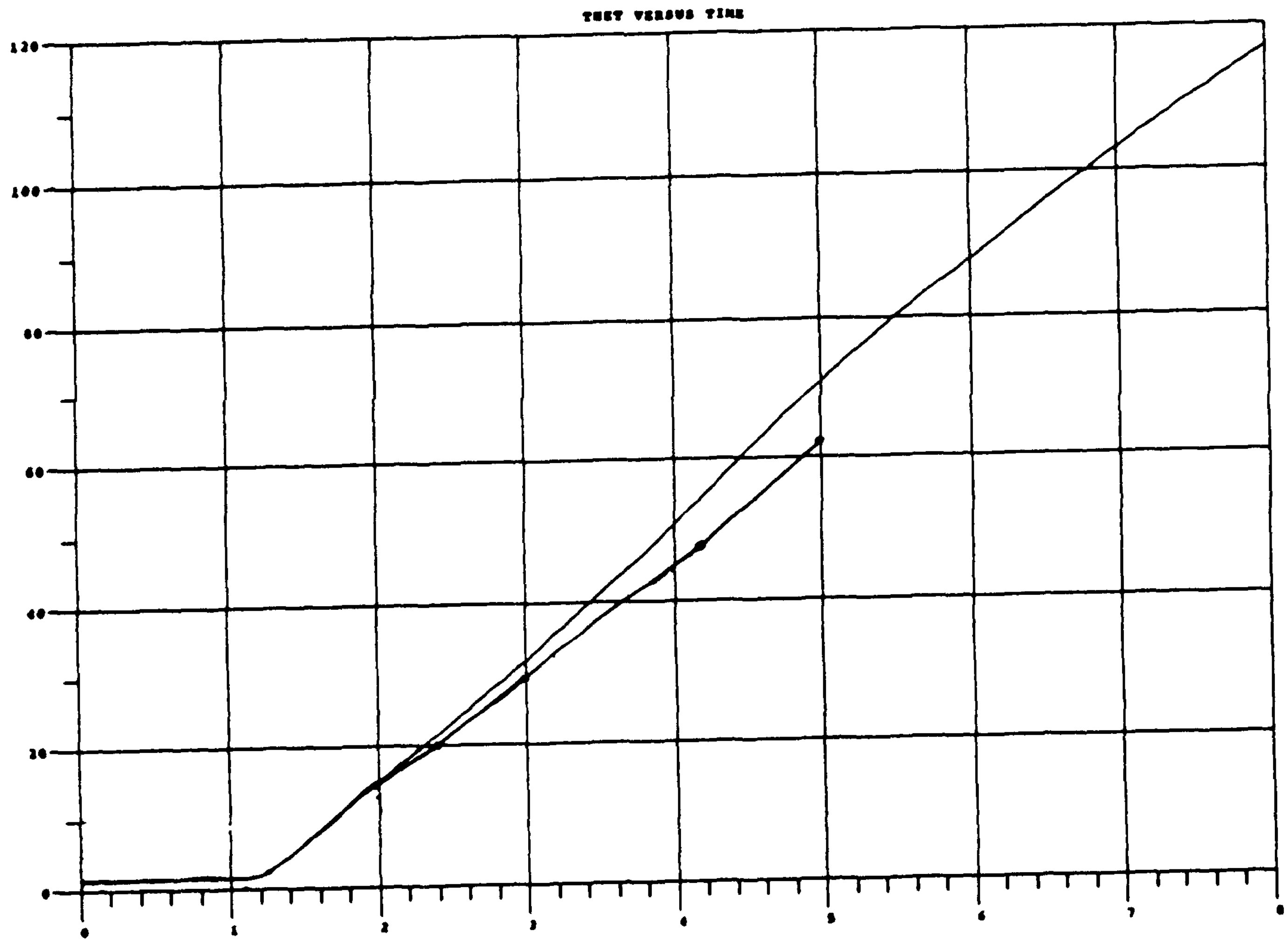
23-APR-91



B.10 Angle of attack vs time for 29 lb longitudinal stick pull and hold for 0.8 Mach 10000 ft check case

F-16 MODEL

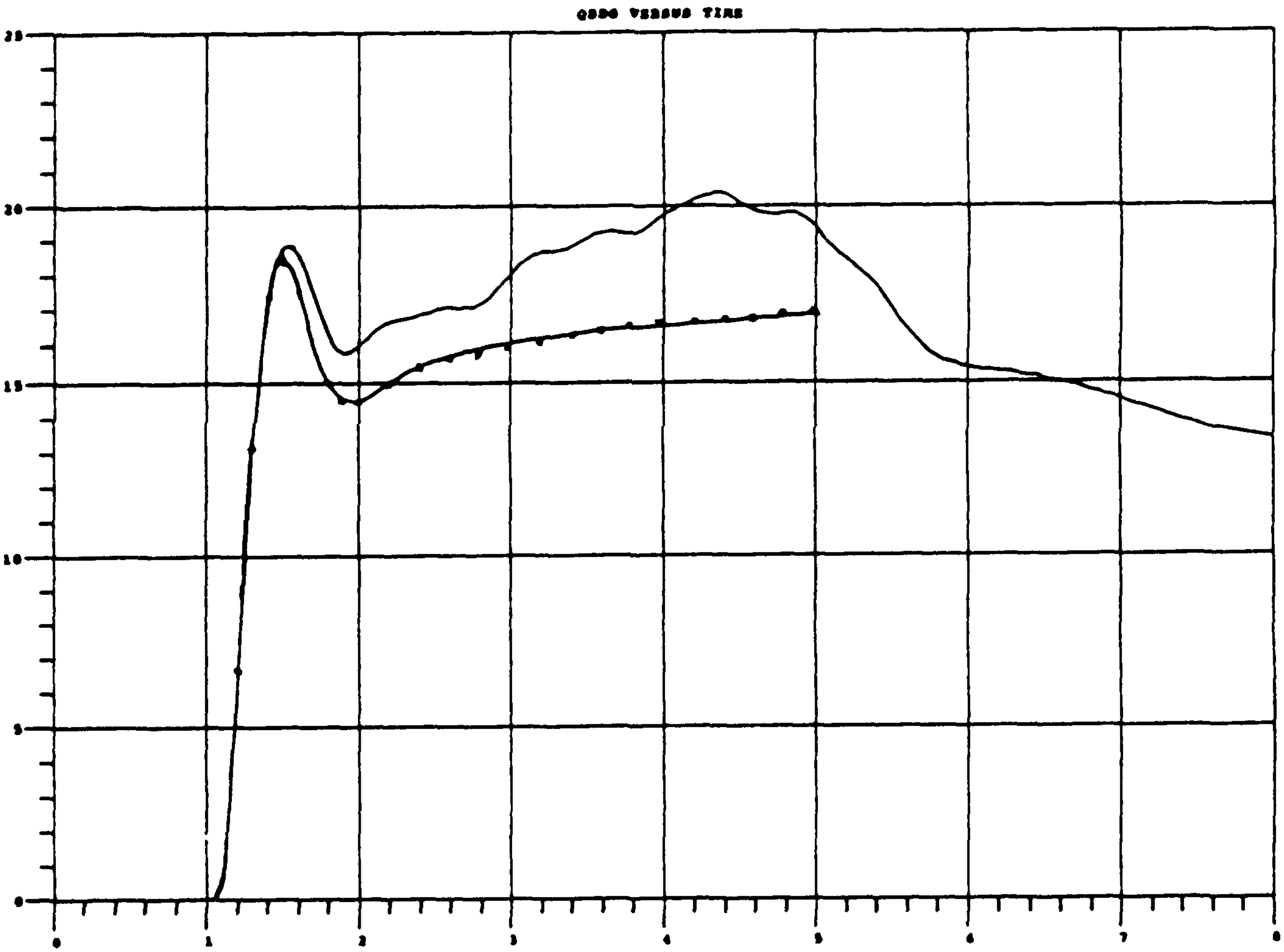
23-APR-91



B.11 Pitch angle vs time for 29 lb longitudinal stick pull and hold for 0.8 Mach 10000 ft check case

F-16 MODEL

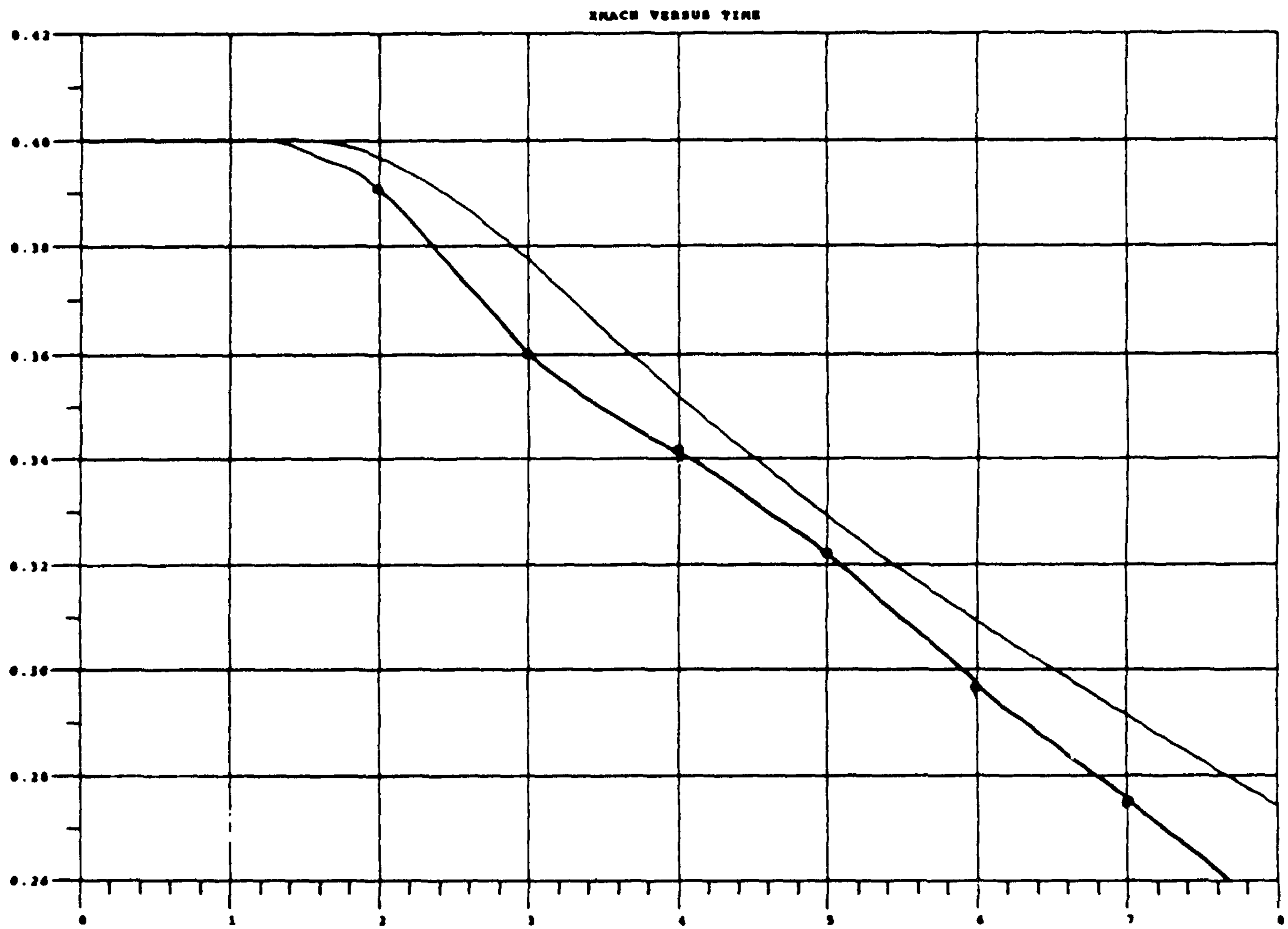
23-APR-91



B.12 Pitch rate vs time for 29 lb longitudinal stick pull and hold for 0.8 Mach 10000 ft check case

F-16 MODEL

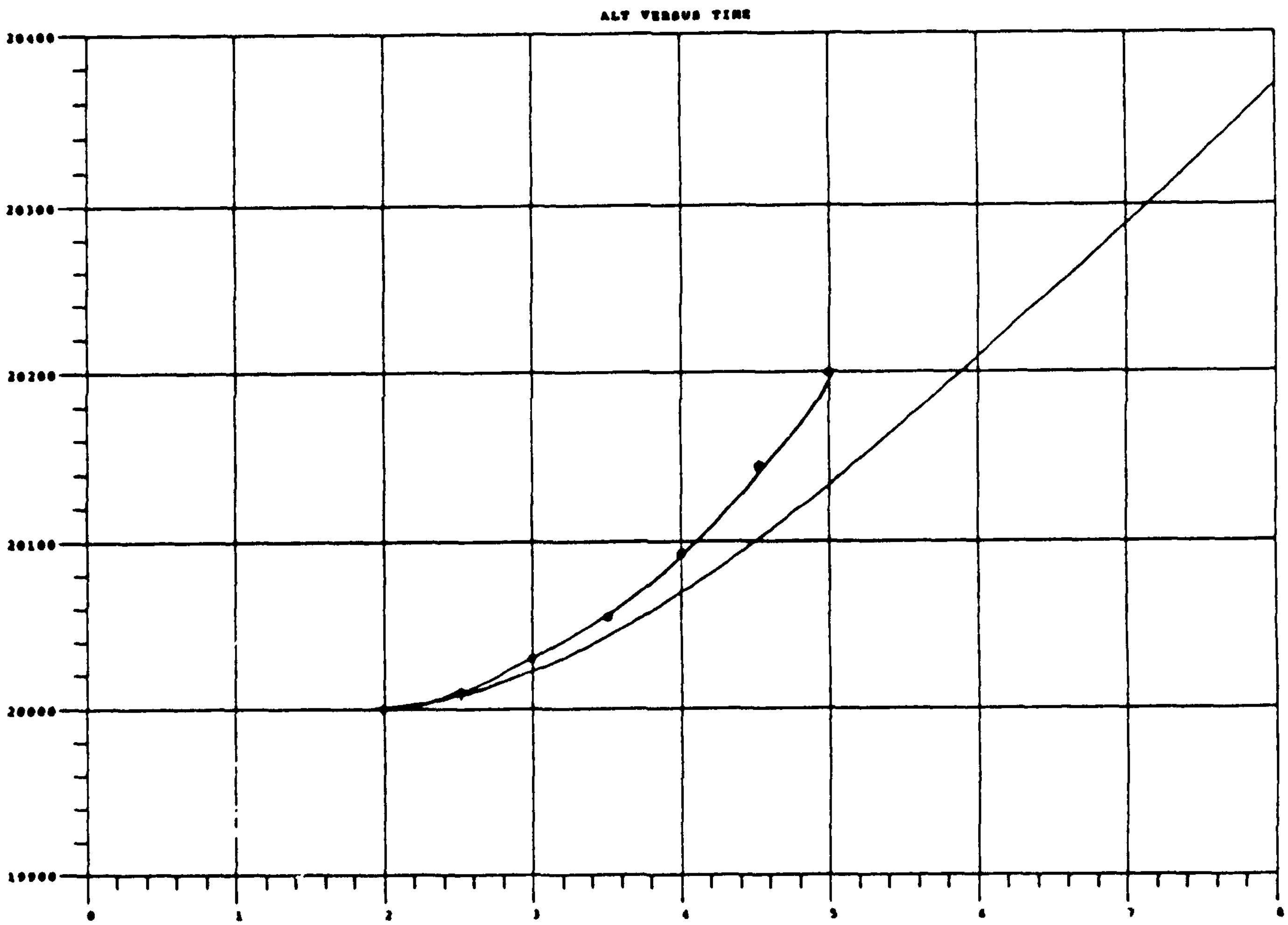
23-APR-91



B.13 Mach vs time for 29 lb longitudinal stick pull and hold for 0.4 Mach 20000 ft check case

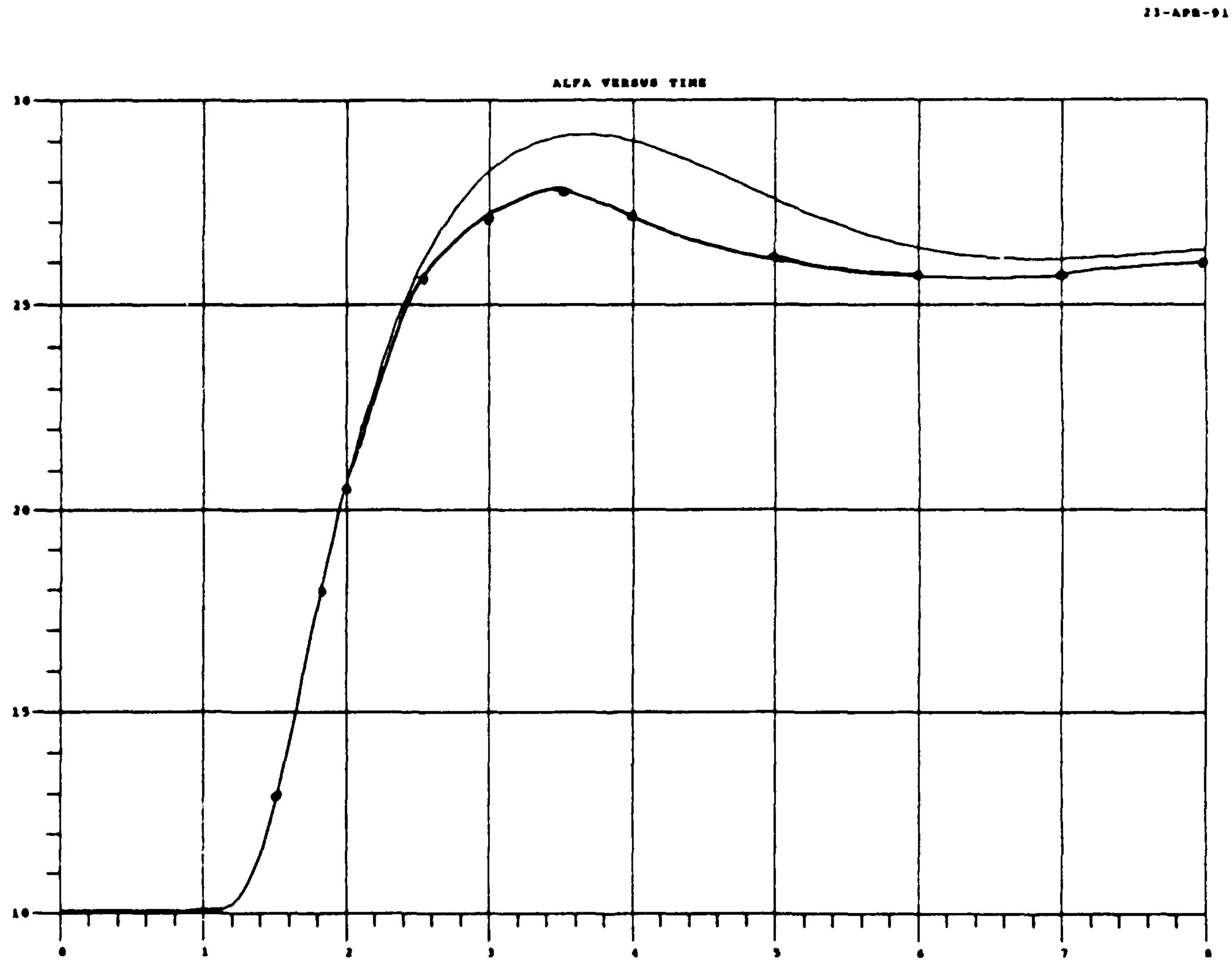
F-16 MODEL

23-APR-91

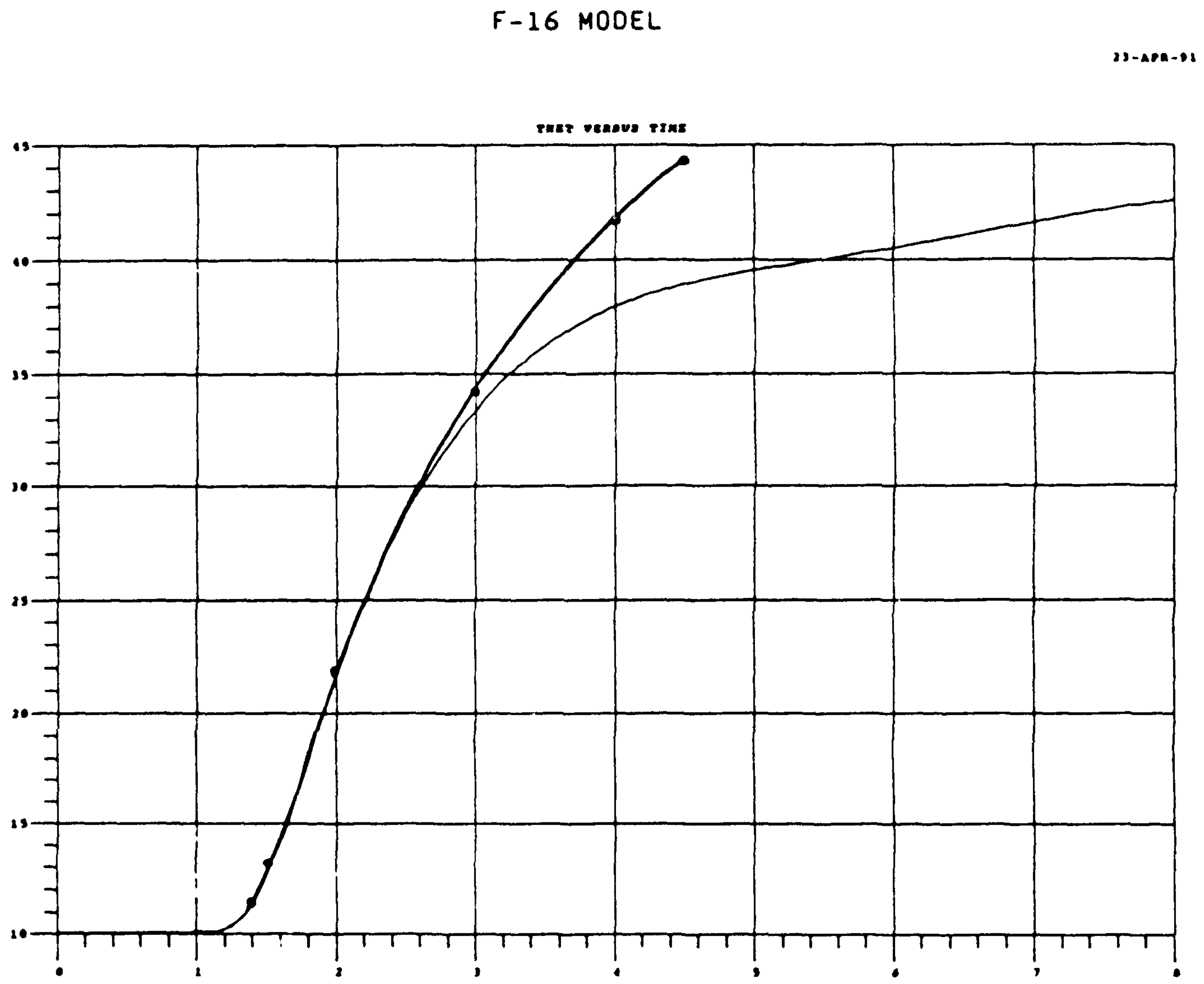


B.14 Altitude vs time for 29 lb longitudinal stick pull and hold for
0.4 Mach 20000 ft check case

F-16 MODEL



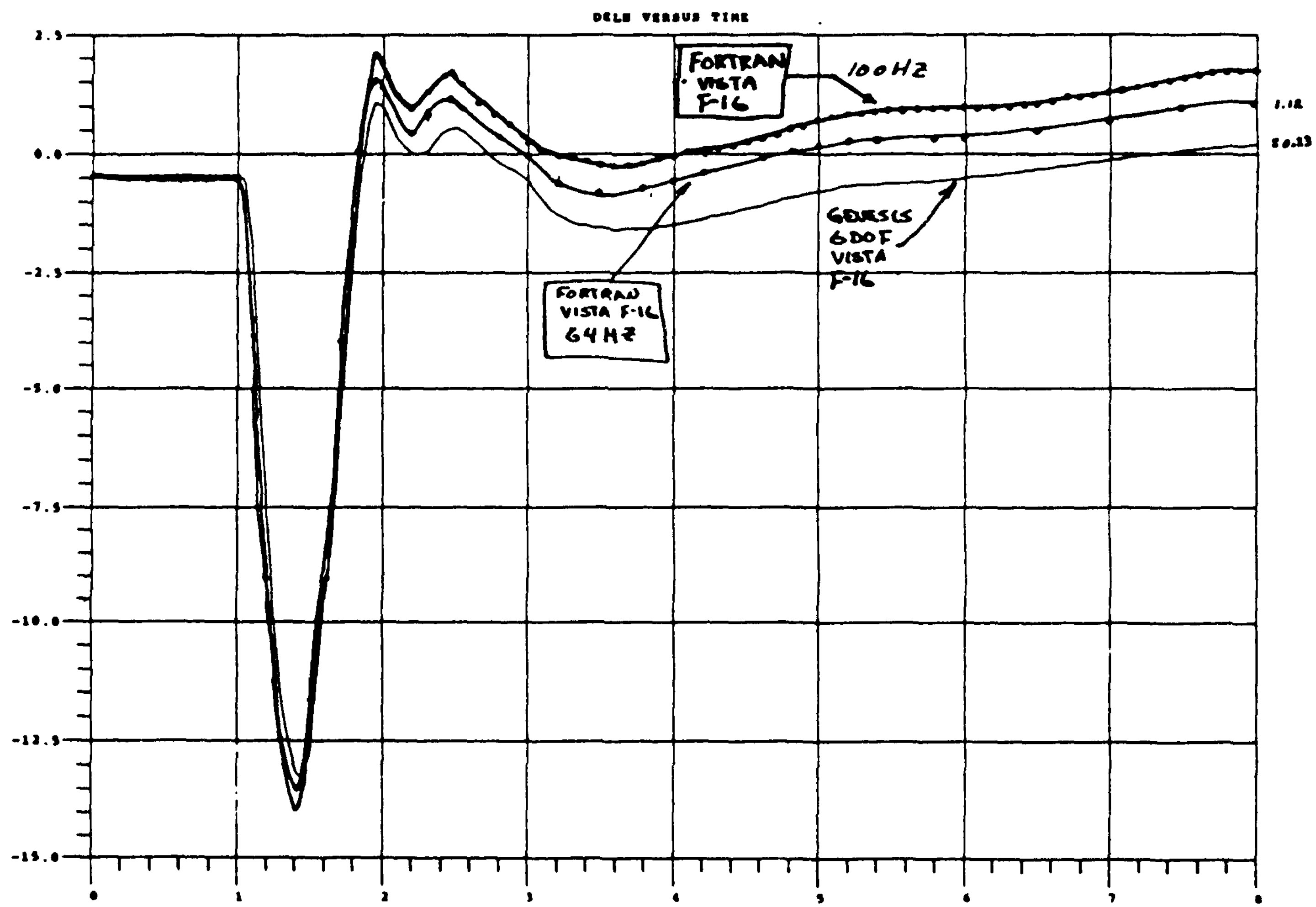
B.15 Angle of attack vs time for 29 lb longitudinal stick pull and hold for 0.4 Mach 20000 ft check case



B.16 Pitch angle vs time for 29 lb longitudinal stick pull and hold for 0.4 Mach 20000 ft check case

F-16 MODEL

23-APR-91



B.17 Horizontal stabilator deflection angle vs time for 29 lb longitudinal stick pull and hold for 0.4 Mach 20000 ft check case

APPENDIX C: MMAESIM COMPUTER CODE

The MMAESIM computer code is divided into subroutines as shown in Figures C.1 - C.8. This appendix will functionally describe each subroutine. The Fortran code is included at the end of this appendix. The appendix will also describe any supporting routines which are necessary or helpful in the execution of this research project.

Figure C.1 presents the MMAESIM fault detection and isolation model key. This figure provides the filter names and their corresponding descriptions. Filters F01XX, F02XX, and F03XX are always the truth models for the three successive time intervals in the simulation (no failure, first failure, second failure). Filter F04XX is always the fully functional filter. The last two letters in the filter designation provide the bank location (B1-B9,and then X0-X3, necessary because of two letter constraint). The MMAESIM program is the first block in Figure C.2. This code is responsible for the proper execution of the subroutines. Figure C.3 demonstrates the block diagrams for the GETDAT and GAUSSGEN subroutines. Figure C.4 presents the Kalman filter subroutine, KFILT, block diagram. Figure C.5 presents the UPDATE and ADPCON subroutine block diagrams. Figure C.6 displays the VISTA F-16 flight control system block diagram (CNTRL). The integration subroutine, DEABM, block diagram is shown in Figure C.7. Figure C.8 presents the equations of motion subroutine, EOM, block diagram. The FORTRAN code is included following the figures. The FORTRAN code for the VISTA F-16 flight control system is not included because of limited distribution rights.

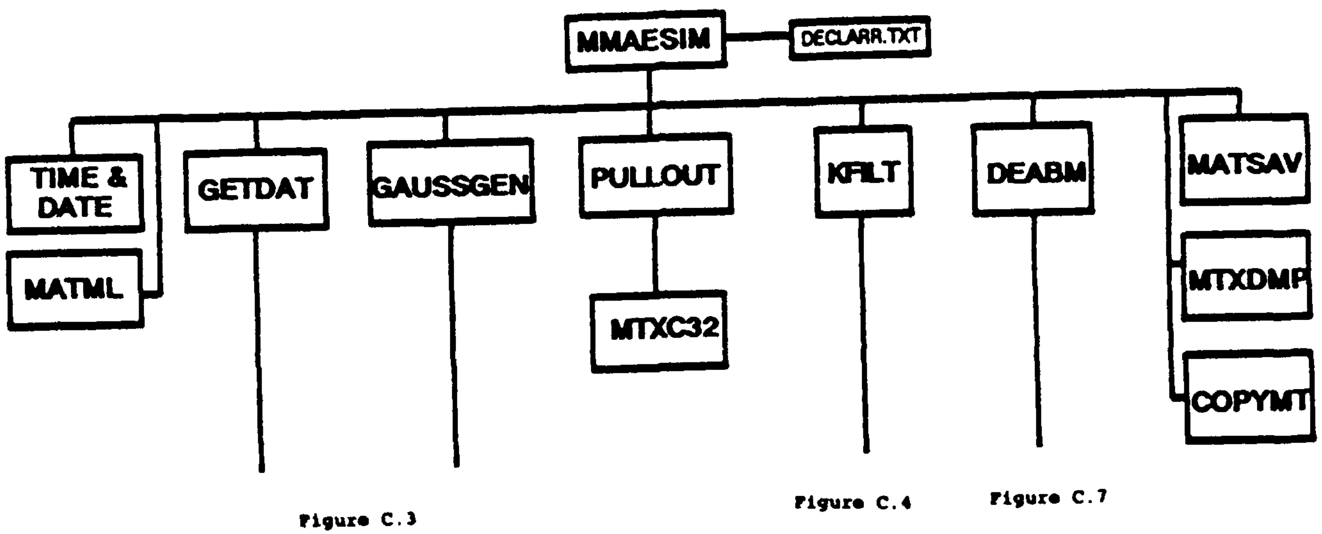
MMAESIM

FAULT DETECTION & ISOLATION MODEL KEY

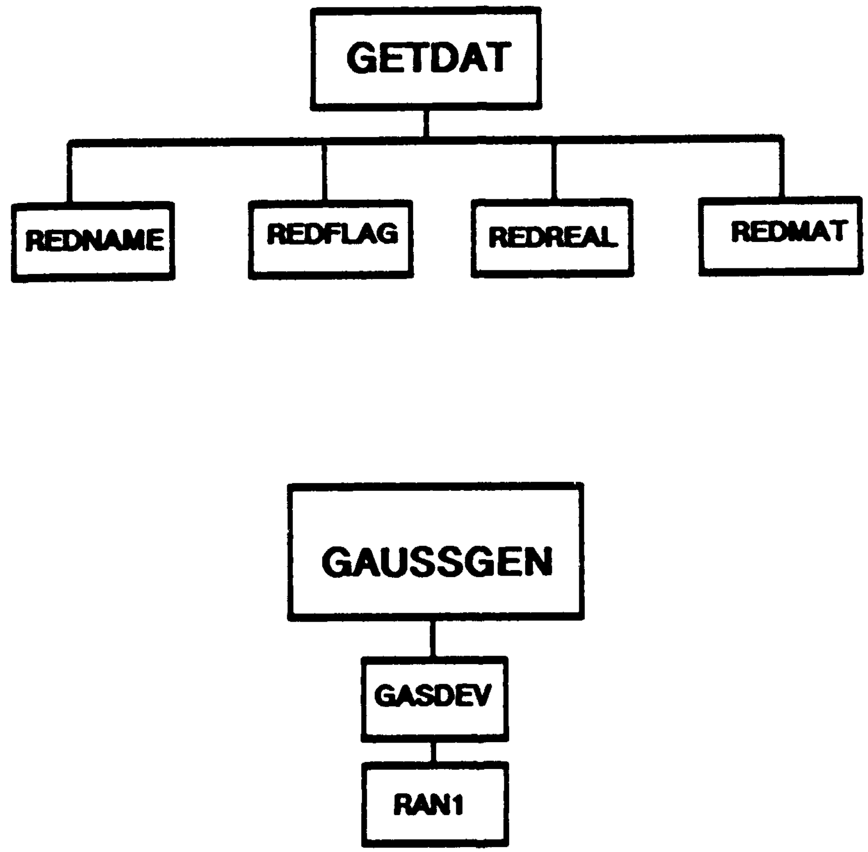
TRUTH	F01B1	– FULLY FUNCTIONAL TRUTH MODEL
	F02B1	– FIRST FAILURE TRUTH MODEL
	F03B1	– DUAL FAILURE TRUTH MODEL
FILTERS	F04B1	– FULLY FUNCTIONAL FILTER
	F05B1	– LEFT STABILATOR FAILURE
	F06B1	– RIGHT STABILATOR FAILURE
	F07B1	– LEFT FLAPERON FAILURE
	F08B1	– RIGHT FLAPERON FAILURE
	F09B1	– RUDDER FAILURE
	F10B1	– VELOCITY SENSOR FAILURE
	F11B1	– ANGLE OF ATTACK SENSOR FAILURE
	F12B1	– PITCH RATE SENSOR FAILURE
	F13B1	– NORMAL ACCELERATION SENSOR FAILURE
	F14B1	– ROLL RATE SENSOR FAILURE
	F15B1	– YAW RATE SENSOR FAILURE
	F16B1	– LATERAL ACCELERATION SENSOR FAILURE

C.1 MMAESIM fault detection and isolation model key

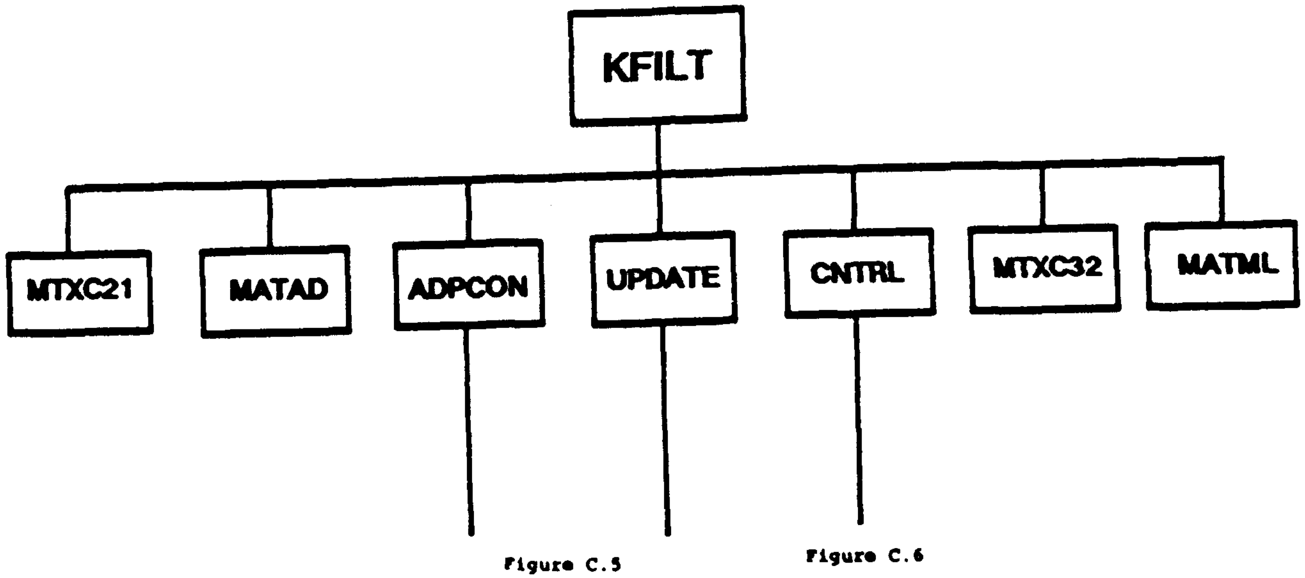
MMAESIM BLOCK DIAGRAM



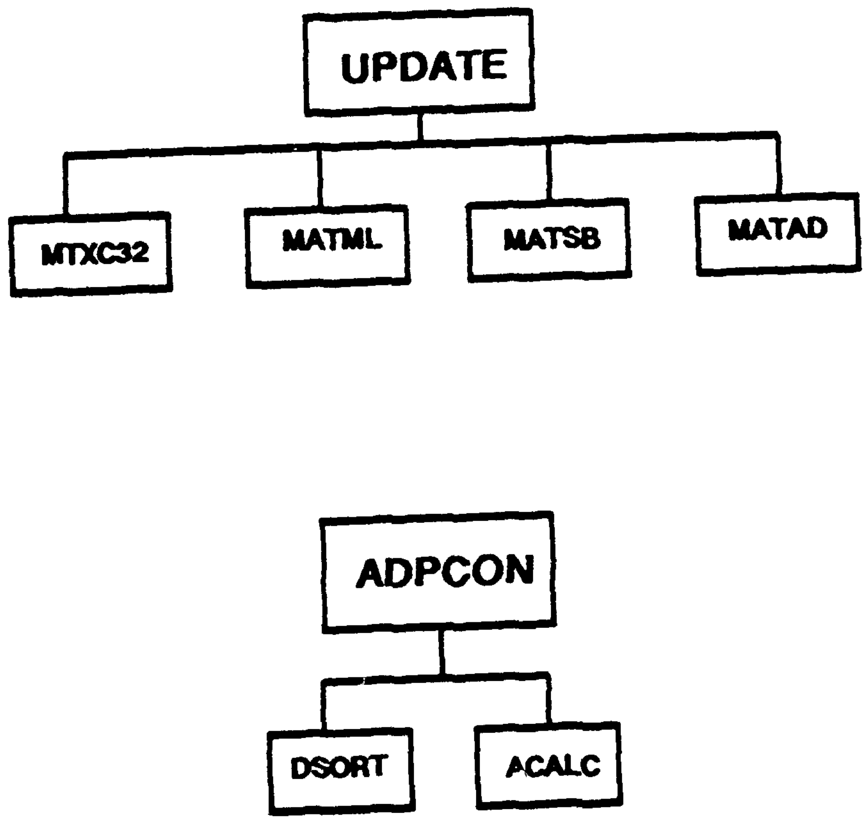
C.2 MMAESIM block diagram



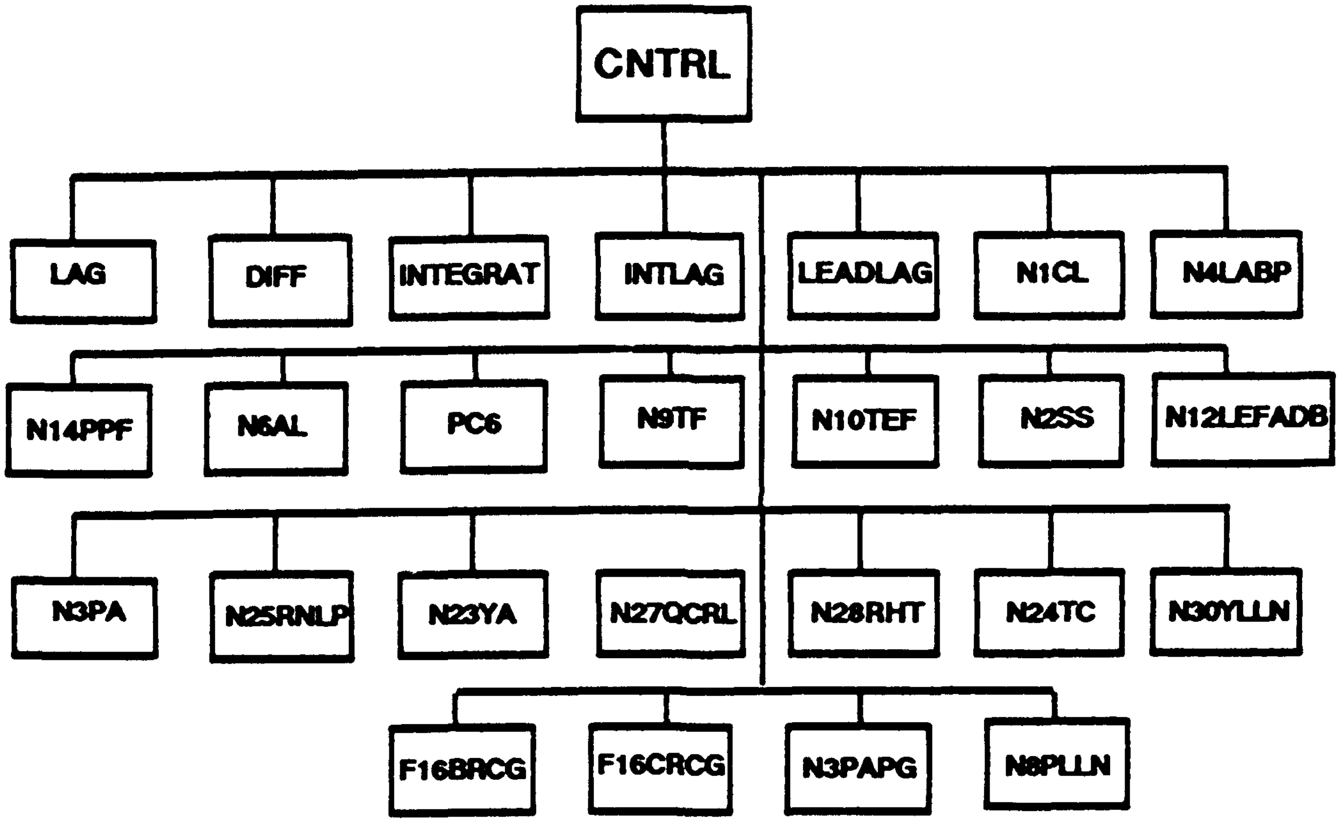
C.3 GETDAT and GAUSSGEN subroutine block diagrams



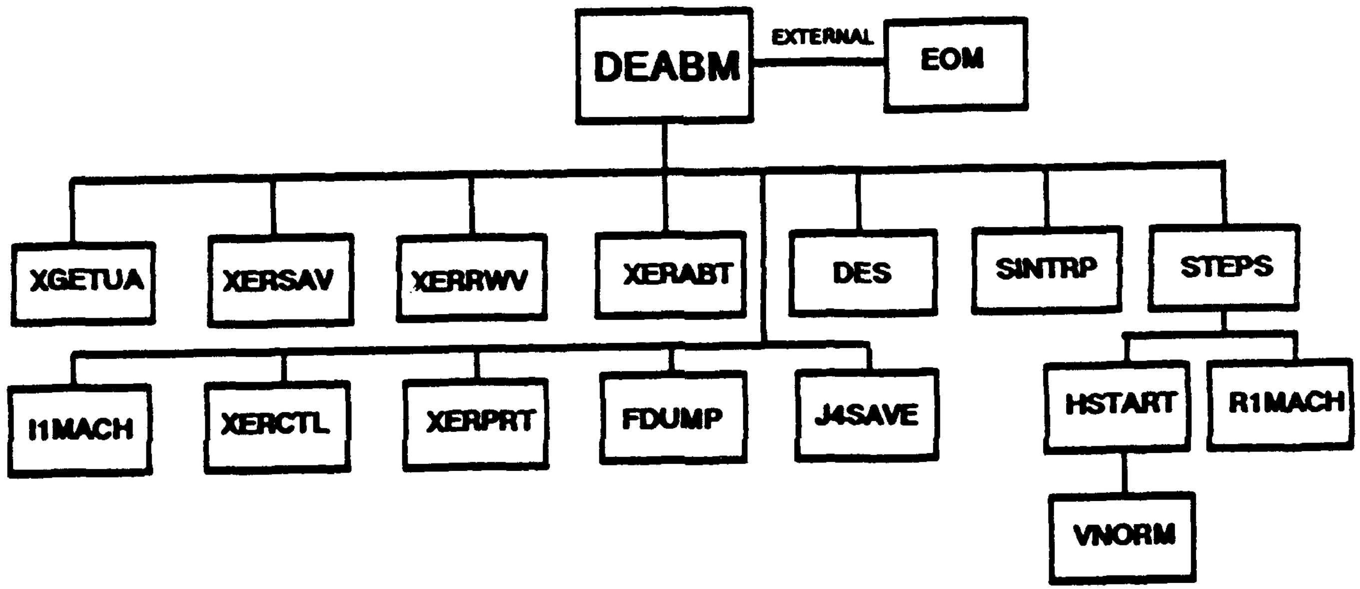
C.4 KFILT subroutine block diagram



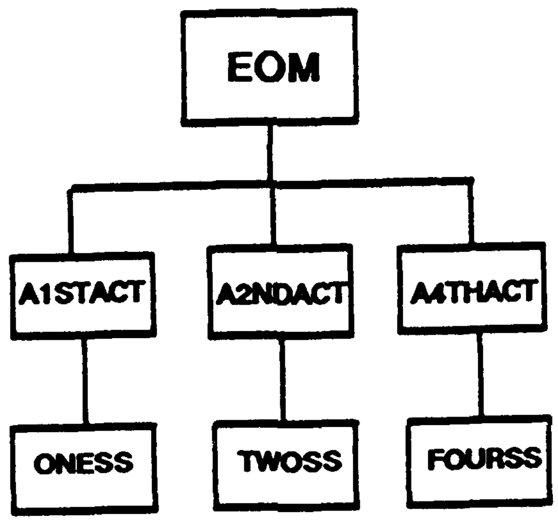
C.5 ADPCON and UPDATE subroutine block diagrams



C.6 CNTRL subroutine block diagram



C.7 DEABM subroutine block diagram



C.8 EOM subroutine block diagram

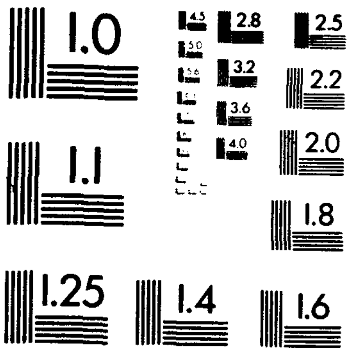
```
      PROGRAM MMAESIM
C*********************************************************************
C                                                                    *
C                AIR FORCE INSTITUTE OF TECHNOLOGY                   *
C         Department of Electrical and Computer Engineering          *
C                                                                    *
C                                                                    *
C                                                                    *
C                                                                    *
C                           EENG 799                                 *
C                                                                    *
C               MULTIPLE MODEL ADAPTIVE ESTIMATOR                    *
C                       FOR THE VISTA/F-16                           *
C                                                                    *
C                                                                    *
C                              by                                    *
C                                                                    *
C                  Captain Gregory L. Stratton                       *
C                              and                                   *
C                         Mr. Tim Menke                              *
C                                                                    *
C                                                                    *
C                                                                    *
C  This program is a simulation to evaluate a multiple model         *
C  adaptive estimator wrapped around the actual General Dynamics     *
C  VISTA/F-16 controller (based on the GD VISTA/F-16 block diagram   *
C  and as coded in FORTRAN by Mr. Tim Menke).  This program is       *
C  based on the program MMACSIM.  The original authors of MMACSIM    *
C  are:                                                              *
C        Captains Donald Pogoda and Gregory Gross, Version 1         *
C        Captain Richard Stevens, Version 3                          *
C                                                                    *
C   Last Revision:     September 1989,       MMAESIM Version:  1     *
C                                                                    *
C*********************************************************************

C  include variable declaration file

      INCLUDE 'DECLARR.TXT'

C . . . . . . . . . . . . . . . . . . . . . .
C       l o c a l   v a r i a b l e s
C . . . . . . . . . . . . . . . . . . . . . .


      REAL  T,TOUT,XIC(29),Y(29),V(7),W(8),NOISE(8)
      REAL  DEGRAD,ZTPART(7),X(29),DX(29)
      INTEGER  I,J,jk,klm
      INTEGER  II,JJ,IX,JX,IZ,JZ,IA,IB,IQ,IDX
      INTEGER  tempat,tempbt,tempct,tempdt
      CHARACTER*1 NFILET,MFILET,OFILET,NYFILET
      CHARACTER*3 NXFILET
      CHARACTER*4 NWFILET
      CHARACTER*5 DFILET

C
      INTEGER L,ti,Bankold,ij,ir,pullflag,count
      REAL Prbavg(20),tswitch,Probs2(10,20)
C
      INTEGER SMPON,SMPOFF
      REAL TIMON,TIMOFF
C
C     declarations for DEABM
C
      INTEGER IDID,IPAR,LIW,LRW,INFO(15),IWORK(50)
      REAL ATOL,RPAR,RTOL,RWORK(739)
```

```
C     declarations for code checking algorithm
C
      REAL hmaydum(7,8),xmaydum(8,1),zmaystat(7,1)
      REAL hmaycon(7,21),xmaycon(21,1),zmaycon(7,1)

C . . . . . . . . . . . . . . . . . . .
      EXTERNAL  EOM
C . . . . . . . . . . . . . . . . . . .

      OPEN(UNIT=17,FILE='TRUTH.DAT',STATUS='UNKNOWN')
C . . . . . . . . . . . . . . . . . . . . . . . . . .

      OPEN(UNIT=18,FILE='CHECKER.DAT',STATUS='UNKNOWN')
C . . . . . . . . . . . . . . . . . . . . . . . . . .

      OPEN(UNIT=19,FILE='FILTER.DAT',STATUS='UNKNOWN')
C . . . . . . . . . . . . . . . . . . . . . . . . . .
      OPEN(UNIT=51,FILE='PROBZZ.DAT',STATUS='UNKNOWN')

C . . . . . . . . . . . . . . . . . . . . . . . . . .
      OPEN(UNIT=52,FILE='PROBZZ2.DAT',STATUS='UNKNOWN')

C . . . . . . . . . . . . . . . . . . . . . . . . . .
      OPEN(UNIT=71,FILE='MXX.DAT',STATUS='UNKNOWN')
C . . . . . . . . . . . . . . . . . . . . . . . . . .
      OPEN(UNIT=72,FILE='MXM.DAT',STATUS='UNKNOWN')
C . . . . . . . . . . . . . . . . . . . . . . . . . .

C-----------------------------------------------------------------------
C                           START PROGRAM
C-----------------------------------------------------------------------

      pi=3.141592654
      degrad=pi/180.
      bankflag=0

      CALL TIME(CTIME)
      CALL DATE(CDATE)

C***********************************************************************
C --- Bring in the data for the truth model(s) and the controllers.
C***********************************************************************

      CALL GETDAT

C***********************************************************************
C --- Compute the stopping sample number for the DSIM provided in
C     the REALS.DAT data input file.
C     Also set the start and stopping times used to store the
C     liklihood information.  Note that only 3 seconds (192 samples
C     based on 64 Hz sample rate) can be saved at a time.  This is
C     to keep the array size of LKH down to a managable level (even
C     as it is, the array is 192 rows x 91 columns = 17,472.  If the
C     given on and off times result in an on and off sample increment
C     difference of greater than 192, then the off sample increment
C     is adjusted so that it is equal to the on sample increment plus
C     192.
C***********************************************************************

      SMPLS=DSIM/TSAMP
      ISTART=1
      ISTOP=IFIX(SMPLS)+1
      TIMON  = 4.0
      TIMOFF = 7.0
      SMPON  = IFIX(TIMON/TSAMP)+1
```

```
      SMPOFF = IFIX(TIMOFF/TSAMP)+1
      IF((SMPOFF-SMPON).GT.192) SMPOFF = SMPON+192

C*********************************************************************
C --- Zero out the storage areas for the states, the inputs,
C     the elemental controller probabilities, the control surface
C     deflections, the output vector, and the accelerations.
C     Each of arrays keeps a running total of their respective values
C     through all of the monte carlo loops.  After the monte carlo
C     loop run is complete, each of the arrays are normalized by the
C     the number of monte carlo loops, XITER.
C     Also zero the time vector used for plotting, TVEC.
C*********************************************************************

      DO 751 JJ=1,10
        DO 750 II=1,513

          IF (JJ.EQ.1) THEN
            TVEC(II,JJ)=0.
            IF (II.LE.192) TSHORT(II,JJ)=0.
          END IF

          IF (JJ.le.8) STATES(II,JJ)=0.

          IF (JJ.le.6) THEN
            INPUTS(II,JJ)=0.
            DEFLEC(II,JJ)=0.
          END IF

          PROBS(II,JJ)=0.
          IF (JJ.LE.7) PROBS(II,JJ+10)=0.

          OUT(II,JJ)=0.
          OUT(II,JJ+10)=0.
          IF (JJ.LE.9) OUT(II,JJ+20)=0.

          IF(JJ.LE.2) ACCEL(II,JJ)=0.

 750    CONTINUE
 751  CONTINUE

C************************************************************************
C  Create the C matrix.  It is just a 29 by 29 identity matrix.
C************************************************************************

      DO JJ=1,29
        DO II=1,29
          IF (JJ.EQ.II) THEN
            C(II,JJ)=1.0
          ELSE
            C(II,JJ)=0.0
          END IF
        END DO
      END DO

C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

C*********************************************************************
C
C  This is the Monte Carlo Simulation Loop, (to statement 780.)
C
C*********************************************************************

      DO 780 IJK=1,XITER
```

AD-A256 569 MULTIPLE MODEL ADAPTIVE ESTIMATION APPLIED TO THE VISTA 2/3
F-16 WITH ACTUATOR AND SENSOR FAILURES VOLUME 2(U) AIR
FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF EN

UNCLASSIFIED T E MENKE 1 JUN 92 AFIT/GA/ENG/92J-01 NL


1.0
4.5
2.8
2.5
5.0
3.2
2.2
5.6
3.6
2.0
4.0
1.1
1.8
1.25
1.4
1.6

```
      WRITE(*,*)' MONTE CARLO LOOP # ',IJK

      INITV=1
      Initv2=0
      MODELN=modeln1

C**********************************************************************
C --- Get the fully functional aircraft truth model matrices.
C**********************************************************************

      CALL PULLOUT

C**********************************************************************
C --- Zero out 29-dimensional X-vector & AUNEW.  Also, initialize
C     the 20-dimensional (seventeen filters plus three truth models.)
C     PRBNEW vector.
C**********************************************************************

      DO 1010 I=1,29
        X(I)=0.0
        IF(I.LE.6)AUNEW(I)=0.0

        IF (I.le.20) THEN
          IF ((I.ne.modeln1) .and. (I.ne.modeln2)
     +                       .and. (I.ne.modeln3)) THEN

            PRBNEW(I,Bank)=(1.-PRBFLTRT0)/FLOAT(NFLTR(Bank)-4)
            IF (I.EQ.FLTRT0) PRBNEW(I,Bank)=PRBFLTRT0

          END IF
        END IF
 1010 CONTINUE

C**********************************************************************
C --- Initialize Differential Equation Solving Routine (DEABM)
C**********************************************************************

      INFO(1)=0
      INFO(2)=0
      INFO(3)=0
      INFO(4)=1
      LRW=739
      LIW=50
C     RTOL=1.E-08
      RTOL=1.E-10
C     ATOL=1.E-07
      ATOL=1.E-09
      T=0.0
      TOUT=0.0

C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

C**********************************************************************
C --- This is the start of the time response loop (to statement 300.)
C**********************************************************************

      DO 300 J=ISTART,ISTOP

C**********************************************************************
C --- Check to make sure we have the correct truth model.
C
C --- If no failures are being modeled (numfails=0) then the fully
C     functional aircraft truth model (modeln1) is always used.
C
C --- If one failure is being modeled (numfails=1) then the fully
C     functional aircraft model (modeln1) is used from time zero to
```

```
C      timelag1.  At timelag1, the truth model being used changes to
C      the single induced failure truth model (modeln2).
C
C --- If two failures are being modeled (numfails=2) then the fully
C      functional aircraft model (modeln1) is used from time zero to
C      timelag1.  At timelag1, the truth model being used changes to
C      the single induced failure truth model (modeln2).  At timelag2,
C      the truth model being used changes to the double induced failure
C      truth model (modeln3).
C
C --- Timelag1 must be less than or equal to timelag2.
C
C --- Note that PULLOUT is called only when MODELN changes.
C**********************************************************************

        IF (numfails.eq.0) THEN
            IF(iactfl2.eq.6)THEN
              IF (tout.ge.timelag2) THEN
                IF (MODELN.ne.modeln3) THEN
                  MODELN=modeln3
                  write(*,*)'truth model = ',MODELn3
                  CALL PULLOUT
                END IF
              END IF
            ENDIF

        ELSE IF (numfails.eq.1) THEN
          IF ((tout.ge.timelag1) .and. (MODELN.ne.modeln2)) THEN
            MODELN=modeln2
            CALL PULLOUT
          END IF

        ELSE IF (numfails.eq.2) THEN
          IF ((tout.ge.timelag1) .and. (tout.lt.timelag2)) THEN
            IF (MODELN.ne.modeln2) THEN
              MODELN=modeln2
              CALL PULLOUT
            END IF
          ELSE IF (tout.ge.timelag2) THEN
            IF (MODELN.ne.modeln3) THEN
              MODELN=modeln3
              write(*,*)'truth model = ',MODELn3
              CALL PULLOUT
            END IF

          END IF

        END IF
C //////////////////////////////////////////
C         CODE CHECK
C //////////////////////////////////////////
C       IF((T.GE.3.0).AND.(T.LE.3.3))THEN
C         WRITE(18,*)'TRUTH MODEL AT TIME',T
C         WRITE(18,*)'HT ',HT
C         WRITE(18,*)'H',H
C         WRITE(18,*)'X',X
C       ENDIF

C**********************************************************************
C --- Create Output Vector, Y. (Y=CX)
C**********************************************************************

        CALL MATML(C,X,Y,29,29,1)

C**********************************************************************
C --- Create the time vector used for plotting purposes, TVEC
```

```
C     And create the short time vector, TSHORT
C**********************************************************************

          IF (IJK.EQ.1) THEN
            TVEC(J,1)=TOUT
            IF ((J.GE.SMPON).AND.(J.LT.SMPOFF)) THEN
              TSHORT(J-SMPON+1,1)=TOUT
            END IF
          END IF

C**********************************************************************
C --- Compute Time Response for System Outputs.
C**********************************************************************

          DO 876 klm=1,29
            OUT(J,klm)=Y(klm)+OUT(J,klm)
 876      CONTINUE

C**********************************************************************
C --- Compute Time Response for Control Inputs and Deflections.
C**********************************************************************

          DO 765 klm=1,6
            INPUTS(J,klm)=AUNEW(klm)+INPUTS(J,klm)
            DEFLEC(J,klm)=X(9+(klm-1)*4)+DEFLEC(J,klm)
 765      CONTINUE

C**********************************************************************
C --- Compute Time Response for Plant States.
C**********************************************************************

          DO 654 klm=1,8
            STATES(J,klm)=X(klm)+STATES(J,klm)
 654      CONTINUE

C**********************************************************************
C --- Compute Time Response of Controller Probabilities.
C**********************************************************************

          DO  iq=1,NFLTR(bank)
            Prbavg(iq)=0.
          ENDDO

          DO 1771 iq=1,NFLTR(Bank)
            IF ((iq.ne.modeln1) .and. (iq.ne.modeln2)
     +                          .and. (iq.ne.modeln3)) THEN

              PROBS(J,iq)=PROBS(J,iq)+PRBNEW(iq,Bank)

              IF(BANK.NE.1)THEN
                PRBBNK2(J,iq)=PROBS(J,iq)
              ENDIF

              IF (J.lt.10) THEN
                DO 1778 ti=2,J
                  Probs2(ti,iq)=Probs2(ti-1,iq)
 1778           CONTINUE
              ELSE
                DO 1779 ti=2,10
                  Probs2(ti,iq)=Probs2(ti-1,iq)
 1779           CONTINUE
              END IF
              Probs2(1,iq)=Prbnew(iq,bank)

              IF (j.gt.10) THEN
                DO 1776 ti=1,10
```

```
                Prbavg(iq)=Prbavg(iq)+Probs2(ti,iq)
 1776         CONTINUE
              Prbavg(iq)=Prbavg(iq)/10.0
            END IF

          END IF

 1771     CONTINUE

C*********************************************************************
C --- Increment the time by one sample time
C*********************************************************************

          TOUT = TOUT + TSAMP

C*********************************************************************
C --- Failure Section - Zero out the failed actuator states for the
C     proper failure at the correct time (currently set up for a
C     single failure scenario
C*********************************************************************

C        write(*,*) tout,numfails,iactfail
          IF ((numfails.eq.1).OR.(numfails.eq.2)) THEN


            IF (tout.ge.timelag1) THEN
              IF (iactfail.eq.1)then
                  X(9)=0.0
                ELSE IF (iactfail.eq.2)then
                  X(13)=0.0
                ELSE IF (iactfail.eq.3)then
                  X(17)=0.0
                ELSE IF (iactfail.eq.4)then
                  X(21)=0.0
                ELSE IF (iactfail.eq.5)then
                  X(25)=0.0
              ENDIF
            ENDIF

            IF (tout.ge.timelag2) THEN
              IF (iactfl2.eq.1)then
                  X(9)=0.0
                ELSE IF (iactfl2.eq.2)then
                  X(13)=0.0
                ELSE IF (iactfl2.eq.3)then
                  X(17)=0.0
                ELSE IF (iactfl2.eq.4)then
                  X(21)=0.0
                ELSE IF (iactfl2.eq.5)then
                  X(25)=0.0
                ELSE IF (iactfl2.eq.6)then
C
C                 conditions have changed no failure is
C                 present.  This demonstrates the capability
C                 of the algorithm to back out of the banks
C
                  numfails = 0
c                  write(*,*)'through gate #1 ',tout

              ENDIF


            ENDIF
          ENDIF
```

```
C**********************************************************************
C --- Create a noise-free measurement vector Z=(HX) and then add
C     measurement noise (+ R) to measurements.  Subroutine Gaussgen
C     generates a GWN vector of length seven.
C
C --- If sensorbias is 1, a sensor bias will be simulated.  The term
C     zbiasamnt is added to the measurement vector.  The sensorbias
C     flag and zbiasamnt vector are defined in the input file.
C**********************************************************************

          CALL MATML(HT,X,ZTPART,7,29,1)

C ////////////////////////////////////////////
C           CODE CHECK
C ////////////////////////////////////////////
C        IF((T.GE.3.0).AND.(T.LE.3.3))THEN
C            WRITE(17,401)T,Z(7)
C            WRITE(18,*)'TRUTH MODEL AT TIME',T
C            WRITE(18,*)'ZTPART = HT * X',ZTPART
C        ENDIF
C ////////////////////////////////////////////


          CALL GAUSSGEN(DSEED,7,V)


          DO 259 IDX=1,7
C
C   Modification to eliminate noise in measurement
C
C           V(IDX)=0.0
C

            Z(IDX)=ZTPART(IDX)
     &           +(SQRT(R(IDX,IDX))*V(IDX))
            IF (tout.ge.timelag1) THEN
              IF (sensorbias.eq.1) Z(IDX)=Z(IDX)+Zbiasamnt(IDX)
            END IF
259      CONTINUE
C ////////////////////////////////////////////
C           CODE CHECK
C ////////////////////////////////////////////
        IF((T.GE.2.95).AND.(T.LE.3.3))THEN
             do imaydum=1,7
               do jmaydum = 1,8
                hmaydum(imaydum,jmaydum)=ht(imaydum,jmaydum)
               enddo
             enddo
             do imaydum = 1,8
                xmaydum(imaydum,1)=x(imaydum)
             enddo
             Call Matml(hmaydum,xmaydum,zmaystat,7,8,1)
             Write(17,*)' State portion of measurement matrix'
             Write(17,*)' time, ht '
             WRITE(17,*)T,hmaydum
             WRITE(17,*)' x ',xmaydum
             WRITE(17,*)' z ',zmaystat

             do imaydum=1,7
               do jmaydum = 1,21
                hmaycon(imaydum,jmaydum)=ht(imaydum,jmaydum + 8)
```

```
               enddo
            enddo
            do imaydum = 1,21
               xmaycon(imaydum,1)=x(imaydum + 8)
            enddo
            write(17,*)' Controller portion of measurement matrix'
            write(17,*)' time, ht'
            Call Matml(hmaycon,xmaycon,zmaycon,7,21,1)
            WRITE(17,*)T,hmaycon
            WRITE(17,*)' x ',xmaycon
            WRITE(17,*)' z ',zmaycon
C           WRITE(18,*)'ZTPART = HT * X',ZTPART
        ENDIF
C ////////////////////////////////////////

C*******************************************************************
C  Save the values of the normal and lateral acceleration
C  into ACCEL so later save with a call to MATSAV.
C  Remember that Z(4) and Z(7) must be adjusted to reflect
C  the true accelerations.
C*******************************************************************

          ACCEL(J,1)=ACCEL(J,1)+Z(4)+1
          ACCEL(J,2)=ACCEL(J,2)+Z(7)-SIN(X(5))

C*******************************************************************
C --- Call the Kalman filter. (ITIME is used to plot the residual
C     time sequence.  See subroutine UPDATE and the residual plotting
C     portion on PLTLSR.)
C*******************************************************************

          itime=j

          CALL KFILT(T,X,DX)

C*******************************************************************
C --- Call the differential equation solver.
C*******************************************************************

          RWORK(1)=TOUT
          write(18,*)'time = ',t,'x before deabm ',x
 1777     CALL DEABM(EOM,29,T,X,TOUT,INFO,RTOL,ATOL,IDID,
     +                RWORK,LRW,IWORK,LIW,RPAR,IPAR)

          INFO(1)=0

          IF (IDID.EQ.3) THEN
            INFO(4)=1

          END IF

          IF (IDID.EQ.(-2)) THEN
            INFO(1)=1
            GOTO 1777
          END IF

          IF ((IDID.LE.0) .AND. (IDID.NE.(-2))) THEN
            PRINT*,'**** ERROR DETECTED WHILE DEABM CALLED ****'

            PRINT*,'  ERROR CODE = ',IDID
            PRINT*,' 500 MORE TIMES'
            INFO(1) =1
            GOTO 1777
C           GOTO 9999
          ELSE
```

```
          INFO(1)=0
          INFO(4)=1

        END IF
        write(18,*)'time = ',t,'x after deabm ',x

        T=TOUT

C*****************************************************************
C --- Corrupt the system with white Gaussian noise if WFLAG=1
C*****************************************************************

        IF (WFLAG.EQ.1) THEN

          CALL GAUSSGEN(DSEED,8,W)

          DO 9098 IX=1,8
            NOISE(IX)=0.
            DO 9995 JX=1,IX
              NOISE(IX)=NOISE(IX)+(CQDCNT(IX,JX)*W(JX)*WGNFAC)
 9995       CONTINUE
            X(IX)=X(IX)+NOISE(IX)
 9098     CONTINUE

        END IF

C*****************************************************************
C     HIERARCHY MODULE
C*****************************************************************
C --- Measure the average of the last 10 samples of the elemental
C     controller probabilities.  If the average is greater than 90%,
C     then declare that failure has occured, and move to the appropriate
C     bank to watch for a second failure, or the full-function aircraft.
C*****************************************************************

C*****************************************************************
C --- Only compute PRBAVG if we are 10 or more samples into the test.
C*****************************************************************

        IF (j.lt.10) GO TO 1773

C*****************************************************************
C--1--First, run through each of the filters in the bank being tested.
C*****************************************************************

        DO 1773 iq=4,NFLTR(Bank)

C*****************************************************************
C--2--If PRBAVG is less than 90%, do not switch banks.
C*****************************************************************

          IF (PRBAVG(iq).lt.0.90) go to 1773

C*****************************************************************
C--3--Do the following if the probability exceeds 90%.
C     Cycle through each of the filter bank names until we find
C     a bankname which is the same as the filter which exceeds
C     the 90% threshold.
C*****************************************************************

          DO 1774 ir=1,numbanks

C*****************************************************************
C --- If BANK equals the bank we are going to test, skip it and
C     proceed to the next bank.
C*****************************************************************
```

```
              IF (Bank.eq.ir) GO TO 1774
              Write(*,*)dfile(iq,bank), Bankname(ir),tout,numfails

c ////////////////////////////////////////////////////////////////////////
c --- This subsection of code converts the bank in dfile to the
c     correct proper bank number for switching banks
c ////////////////////////////////////////////////////////////////////////

              DFILET = dfile(iq,bank)

              NFILET=DFILET(2:2)
              MFILET=DFILET(3:3)

              TEMPAT = ICHAR(NFILET) - 48
              TEMPBT = ICHAR(MFILET) - 48
              TEMPCT = TEMPAT*10 + TEMPBT
              TEMPDT = TEMPCT

              IF(TEMPCT.LE.12)THEN
                TEMPCT = TEMPCT - 3
              ELSE
                TEMPCT = TEMPCT - 3
              ENDIF

              WRITE(*,*)NFILET,MFILET,TEMPAT,TEMPBT,TEMPCT

c              NWFILET = DFILET
c              NXFILET = DFILET
c              OFILET = CHAR(TEMPCT + 48)
c              WRITE(*,*)OFILET
c
c              IF(TEMPDT.LE.12)THEN
c                dfilet = nwfilet//ofilet
c              ELSE
c                nyfilet = 'x'
c                dfilet = nxfilet//nyfilet//ofilet
c              ENDIF
c              write(*,*)dfilet

c ///////////////////////////////////////////////////////////////////

              IF (DFILE(iq,tempct).eq.Bankname(ir)) THEN
                Bankold=Bank
                Bank=ir
                Bankflag=Bankflag+1
                PRINT*,' '
                WRITE(*,9700) Dfile(iq,bankold),Dfile(iq,bank),tout
 9700           FORMAT('$',5x,'We are switching from bank ',A,' to
     +          bank ',A,' at time ',F4.2,'.')
                PRINT*,' '
                PRINT*,' '

c ////////////////////////////////////////////////////////////////////////
c
c --- Experimental
c
c ////////////////////////////////////////////////////////////////////////

                IF(Bank.gt.1)then

                  Do i = 1, Numbanks
                   Bankname(i) = Bankname(1)
                  Enddo
```

```
              ELSE

                Do i = 1, Numbanks
                 Bankname(i) = Dfile((i+3),i)
                Enddo

              ENDIF

C**********************************************************************
C--4--Set the probability value of the new filter equal to the
C     probability of the filter in the previous bank.  Also set
C     the other filter probabilities equally so the overall
C     probability equals one.
C**********************************************************************

              DO 1781 ti=4,NFLTR(bank)
                IF (ti.eq.iq) THEN
                  prbnew(ti,ir)=prbnew(iq,bankold)
                ELSE
                  prbnew(ti,ir)=(1.0-prbnew(iq,bankold))/
     &                          (float(NFLTR(bank)-4))
                ENDIF
 1781         CONTINUE
              tswitch=tout/(tsamp)

C**********************************************************************
C--5--This test tells us if we have tested all banks but cannot
C     locate an appropriate bank.
C**********************************************************************

            ELSE IF ((ir.eq.numbanks) .and. (count.eq.0)) THEN
              WRITE(*,9701)
 9701         FORMAT('$',5x,'We have passed the switch test,
     + but have no other bank to switch to, ')
              WRITE(*,9703) Bank
 9703         FORMAT('$',5x,'so we are staying in bank number',I2,'.')
              WRITE(*,9702) Dfile(4,Bank)
 9702         FORMAT('$',5x,'(The name of this bank is ',A,'.)')
              PRINT*,' '
              count=count+1
            END IF

 1774       CONTINUE

 1773     CONTINUE

C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
      IF((TOUT.GE.0.0).AND.(TOUT.LE.8.0))THEN
         WRITE(51,*)' PRBNEW at time: ',TOUT
         WRITE(52,*)' PRBNEW at time: ',TOUT
       DO IZXQ = 4,16,4
         WRITE(51,*)PRBNEW(IZXQ,1),PRBNEW(IZXQ+1,1),PRBNEW(IZXQ+2,1),
     &   PRBNEW(IZXQ+3,1)

         write(52,*)PRBNEW(IZXQ,2),PRBNEW(IZXQ+1,2),PRBNEW(IZXQ+2,2),
     &   PRBNEW(IZXQ+3,2)

       ENDDO
      ENDIF
C/////////////////////////////////////////////////////////////////////////
C/////////////////////////////////////////////////////////////////////////

 300    CONTINUE

C//////////////////////////// END TIME RESPONSE LOOP ///////////////////////
```

```
C//////////////////////////////////////////////////////////////////////////
        IF (ijk.ne.xiter) THEN
          pullflag=0
          Count=0
          Bank=1
          Bankold=Bank
          Bankflag=0
        END IF

 780  CONTINUE

C$$$$$$$$$$$$$$$$$$$$$$$$$$$ END MONTE CARLO LOOP $$$$$$$$$$$$$$$$$$$$$$$$
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

C**********************************************************************
C --- Now normalize everything by the number of Monte Carlo iterations.
C     And set-up array for single scalar residuals to be read in
C     MATSAV.  The array RSID is set up such that the rows correspond
C     to the time increment number.  The columns are as follows:
C     columns 1:7  correspond to the scalar residuals of filter #4,
C     columns 8:14 correspond to the scalar residuals of filter #5,
C     etc., until the last filter.
C**********************************************************************

      DO 755 JZ=1,10

        DO 754 IZ=1,ISTOP

          IF (JZ.le.8) STATES(IZ,JZ)=STATES(IZ,JZ)/XITER

          IF (JZ.le.6) THEN
            INPUTS(IZ,JZ)=INPUTS(IZ,JZ)/XITER
            DEFLEC(IZ,JZ)=DEFLEC(IZ,JZ)/XITER
          END IF

          PROBS(IZ,JZ)=PROBS(IZ,JZ)/XITER
          IF (JZ.le.7) PROBS(IZ,JZ+10)=PROBS(IZ,JZ+10)/XITER

          OUT(IZ,JZ)=OUT(IZ,JZ)/XITER
          OUT(IZ,JZ+10)=OUT(IZ,JZ+10)/XITER
          IF (JZ.LE.9) OUT(IZ,JZ+20)=OUT(IZ,JZ+20)/XITER

          IF (JZ.LE.2) ACCEL(IZ,JZ)=ACCEL(IZ,JZ)/XITER

          IF (JZ.LE.7)THEN
            IF ((IZ.GE.SMPON).AND.(IZ.LT.SMPOFF))THEN
              DO II=4,NFLTR(1)
                IIMOD=II-3
                RSID(IZ-SMPON+1,INT((II-4)*7+JZ))=rssave(IZ,JZ,IIMOD)
                BDUSG(IZ-SMPON+1,INT((II-4)*7+JZ))=buzsave(IZ,JZ,IIMOD)

                IF(BANK.NE.1)THEN
                  RSIDTWO(IZ-SMPON+1,INT((II-4)*7+JZ))=
     &              rssave(IZ,JZ,IIMOD)
                  BDUSGTWO(IZ-SMPON+1,INT((II-4)*7+JZ))=
     &              buzsave(IZ,JZ,IIMOD)
                ENDIF

C                write(*,*) '---------  mmaesim  -------- ',IZ
C                write(*,*)buzsave(IZ,4,8)
C                iizztpc=IZ-SMPON+1
C                iizztpb=INT((II-4)*7+JZ)
C                write(*,*)BDUSG(iizztpc,iizztpb),iizztpc,iizztpb
```

```
C               if(iizztpb.eq.91)then
C               write(*,*)iizztpc,BDUSG(iizztpc,91)
C               endif

             END DO
           END IF
         END IF

 754    CONTINUE
 755  CONTINUE

C***********************************************************************
C --- Mean and Standard Deviation of Probabilities Compuation
C     Note, the data used from this section should be used with care
C     if hierarchical (2 failures) modeling is used.  For no failures
C     it is assumed that the mean and standard deviation can be
C     averaged over the entire length of the simulation.  For single
C     failures (w/o hierarchical modeling), it is assumed that the
C     mean and standard deviation are averaged over two time periods:
C     (1) from time zero to timelag1, and (2) from timelag1 to the
C     end of the simulation.  For double failures (hierarchical
C     modeling), it is assumed that the mean and standard deviation
C     are average over three time periods:  (1) from time zero to
C     timelag1, (2) from timelag1 to timelag2, and (3) from timelag2
C     to the end of the simulation.  NOTE:  For hierarchical modeling
C     we are assuming the banks are switched at timelag1.  However,
C     there should be a slight lag from timelag1 to bank switching.
C     Therefore, caution should be used when hierarchical modeling
C     is implemented.
C***********************************************************************

      DO kk=1,3
        DO kj=1,NFLTR(1)
          meanprob(kk,kj)=0.0
          stddev(kk,kj)=0.0
        END DO
      END DO

C***********************************************************************
C --- First compute the mean.
C***********************************************************************

      DO kj=4,NFLTR(1)

          IF(numfails.eq.0)THEN
            Temp1=0.0
            DO ki=1,ISTOP
              Temp1=Temp1+Probs(ki,kj)
            END DO
            Meanprob(1,kj)=Temp1/Float(ISTOP)

          ELSE IF(numfails.eq.1)THEN
            Temp1=0.0
            DO ki=1,IFIX(timelag1/tsamp)-1
              Temp1=Temp1+Probs(ki,kj)
            END DO
            Meanprob(1,kj)=Temp1/((timelag1/tsamp)-1.)
            Temp1=0.0
            DO ki=IFIX(timelag1/tsamp),ISTOP
              Temp1=Temp1+Probs(ki,kj)
            END DO
            Meanprob(2,kj)=
     &            Temp1/(Float(ISTOP)-(timelag1/tsamp)+1.)

          ELSE IF(numfails.eq.2)THEN
            Temp1=0.0
```

```
            DO ki=1,IFIX(timelag1/tsamp)-1
              Temp1=Temp1+probs(ki,kj)
            END DO
            Meanprob(1,kj)=Temp1/((timelag1/tsamp)-1.)
            Temp1=0.0
            DO ki=IFIX(timelag1/tsamp),IFIX(timelag2/tsamp)-1
              Temp1=Temp1+probs(ki,kj)
            END DO
            Meanprob(2,kj)=
     &          Temp1/((timelag2-timelag1)/tsamp)
            Temp1=0.0
            DO ki=IFIX(timelag2/tsamp),ISTOP
              Temp1=Temp1+probs(ki,kj)
            END DO
            Meanprob(3,kj)=
     &          Temp1/(Float(ISTOP)-(timelag2/tsamp)+1.)

          END IF
        END DO

C*********************************************************************
C --- Now compute the standard deviation.
C*********************************************************************

        DO kj=4,NFLTR(1)

          IF(numfails.eq.0)THEN
            Temp2=0.0
            DO ki=1,ISTOP
              Temp2=Temp2+((probs(ki,kj)-Meanprob(1,kj))**2)
            END DO
            Stddev(1,kj)=sqrt(Temp2/Float(ISTOP))

          ELSE IF(numfails.eq.1)THEN
            Temp2=0.0
            DO ki=1,IFIX(timelag1/tsamp)-1
              Temp2=Temp2+((probs(ki,kj)-Meanprob(1,kj))**2)
            END DO
            Stddev(1,kj)=sqrt(Temp2/((timelag1/tsamp)-1.))
            Temp2=0.0
            DO ki=IFIX(timelag1/tsamp),ISTOP
              Temp2=Temp2+((probs(ki,kj)-Meanprob(2,kj))**2)
            END DO
            Stddev(2,kj)=
     &         sqrt(Temp2/(Float(ISTOP)-(timelag1/tsamp)+1.))

          ELSE IF(numfails.eq.2)THEN
            Temp2=0.0
            DO ki=1,IFIX(timelag1/tsamp)-1
              Temp2=Temp2+((probs(ki,kj)-Meanprob(1,kj))**2)
            END DO
            Stddev(1,kj)=sqrt(Temp2/((timelag1/tsamp)-1.))
            Temp2=0.0
            DO ki=IFIX(timelag1/tsamp),IFIX(timelag2/tsamp)-1
              Temp2=Temp2+((probs(ki,kj)-Meanprob(2,kj))**2)
            END DO
            Stddev(2,kj)=
     &          sqrt(Temp2/((timelag2-timelag1)/tsamp))
            Temp2=0.0
            DO ki=IFIX(timelag2/tsamp),ISTOP
              Temp2=Temp2+((probs(ki,kj)-Meanprob(3,kj))**2)
            END DO
            Stddev(3,kj)=
     &          sqrt(Temp2/(Float(ISTOP)-(timelag2/tsamp)+1.))

          END IF
```

```
      END DO

C*********************************************************************
C --- Call MATSAV to generate MATRIXX data files for plotting
C*********************************************************************

       CALL MATSAV(71,'TI',513,ISTOP,1,0,TVEC,DUMMYJ,
     &'(10A8)')
       CALL MATSAV(71,'IN',513,ISTOP,6,0,INPUTS,DUMMYJ,
     &'(10A8)')
       CALL MATSAV(71,'ST',513,ISTOP,8,0,STATES,DUMMYJ,
     &'(10A8)')
       CALL MATSAV(71,'DF',513,ISTOP,6,0,DEFLEC,DUMMYJ,
     &'(10A8)')
       CALL MATSAV(71,'PRB',513,ISTOP,17,0,PROBS,DUMMYJ,
     &'(10A8)')
       CALL MATSAV(71,'GS',513,ISTOP,2,0,ACCEL,DUMMYJ,
     &'(10A8)')
       CALL MATSAV(71,'TS',192,SMPOFF-SMPON,1,0,TSHORT,DUMMYJ,
     &'(10A8)')
       CALL MATSAV(71,'RS',192,SMPOFF-SMPON,91,0,RSID,DUMMYJ,
     &'(10A8)')
       CALL MATSAV(71,'BDU',192,SMPOFF-SMPON,91,0,BDUSG,DUMMYJ,
     &'(10A8)')
       CALL MATSAV(71,'MN',3,3,16,0,MEANPROB,DUMMYJ,
     &'(10A8)')
       CALL MATSAV(71,'SIG',3,3,16,0,STDDEV,DUMMYJ,
     &'(10A8)')

C *********************************************************
C  2 nd MATRIXx Plotting file for second bank
C *********************************************************


       CALL MATSAV(72,'TI',513,ISTOP,1,0,TVEC,DUMMYJ,
     &'(10A8)')
       CALL MATSAV(72,'PRB',513,ISTOP,17,0,PRBBNK2,DUMMYJ,
     &'(10A8)')
       CALL MATSAV(72,'TS',192,SMPOFF-SMPON,1,0,TSHORT,DUMMYJ,
     &'(10A8)')
       CALL MATSAV(72,'RS',192,SMPOFF-SMPON,91,0,RSIDTWO,DUMMYJ,
     &'(10A8)')
       CALL MATSAV(72,'BDU',192,SMPOFF-SMPON,91,0,BDUSGTWO,DUMMYJ,
     &'(10A8)')

 9999 CONTINUE

C/////////////////////////////////////////////////////////////////////
C                               END OF MMAESIM
C--------------------------------------------------------------------
C--------------------------------------------------------------------
      END
```

```
      SUBROUTINE ADPCON(PRBTMP,PRBOLD,ZXHP)

      INCLUDE 'DECLARR.TXT'

C . . . . . . . . . . . . . . . . . . .
C  l o c a l   v a r i a b l e s
C . . . . . . . . . . . . . . . . . . .

      INTEGER I,J,K,PORDER(20,15)
      REAL PRBTMP(20,15),PRBOLD(20,15),ZXHP(8,20)
      REAL PRBSUM,PDUMMY(20,15),TEMP,PRBDLT,PRBDUM(20,15)
      REAL*16 PRBWRK,QDELT

C______________________________________________________________________


C*********************************************************************
C --- Calculate the normalizing factor for the probabilities,
C     the denominator of equation 10-104 of Maybeck.
C*********************************************************************

      PRBSUM=0.
      QDELT=0.

      DO 10 I=1,NFLTR(Bank)
        IF ((I.ne.modeln1) .and. (I.ne.modeln2)
     +                     .and. (I.ne.modeln3)) THEN
          PRBSUM=PRBSUM+PRBTMP(I,Bank)
        END IF
 10   CONTINUE

C*********************************************************************
C --- Calculate the probabilities, stored in PRBWRK.
C
C --- If a probability is less than the minimum acceptable
C     probability, PRBMIN, set it equal to PRBMIN.
C
C     Having to reset any probabilities to PRBMIN will cause the sum
C     of the probabilities to exceed one.  To fix this, the following
C     algorithm keeps track of the errors, which is the sum of all the
C     (PRBMIN - PRBWRK).  This error is then subtracted from the highest
C     probability, as found from a call to DSORT.  This results in the
C     sum of the probabilities equal to one.
C
C     QEXT is an intrinsic function that returns .........
C     SNGLQ is a ............
C*********************************************************************

      DO 20 J=1,NFLTR(Bank)
        IF ((J.ne.modeln1) .and. (J.ne.modeln2)
     +                     .and. (J.ne.modeln3)) THEN

          PRBWRK=QEXT(PRBTMP(J,Bank))/QEXT(PRBSUM)

          IF (PRBWRK.LT.QEXT(PRBMIN)) THEN
            QDELT=QDELT+(QEXT(PRBMIN)-PRBWRK)
            PRBWRK=QEXT(PRBMIN)
          END IF

          PRBNEW(J,Bank)=SNGLQ(PRBWRK)
          PRBOLD(J,Bank)=PRBNEW(J,Bank)

        END IF
 20   CONTINUE

      PRBDLT=SNGLQ(QDELT)
```

```
C*********************************************************************
C --- Now, create a vector with the indices of the sorted probabilities
C     (high to low.)  Note:  PRBNEW's ordering is not changed.
C*********************************************************************

      DO I=1,NFLTR(BANK)
         PRBDUM(I,BANK)=PRBNEW(I,BANK)
      ENDDO

      CALL DSORT(PRBDUM,NFLTR,PORDER)

C*********************************************************************
C --- Decrease the highest probability by PRBDLT to ensure overall
C     probability is 1.0
C*********************************************************************

      PRBNEW(PORDER(4,Bank),Bank)=PRBNEW(PORDER(4,Bank),Bank)-PRBDLT
      PRBDUM(PORDER(4,Bank),Bank)=PRBDUM(PORDER(4,Bank),Bank)-PRBDLT

C*********************************************************************
C --- Below are various methods to use the probabilities to calculate
C     the weighted sum state estimate, XHPSUM.  Selection is based on
C     what the flag ISCLT is set to.
C*********************************************************************

C111111111111111111111111111111111111111111111111111111111111111111111
C --- Method 1.  Use only the filter estimate and raw probability
C                of the highest probability filter.

      IF (ISLCT.EQ.1) THEN
        CALL ACALC(ZXHP,PRBDUM,PORDER,4)

C222222222222222222222222222222222222222222222222222222222222222222222
C --- Method 2.  Use only the filter estimate of the highest probability
C                filter and then set that probability equal to one.

      ELSE IF (ISLCT.EQ.2) THEN
        PDUMMY(PORDER(4,Bank),Bank)=1.
        CALL ACALC(ZXHP,PDUMMY,PORDER,4)

C333333333333333333333333333333333333333333333333333333333333333333333
C --- Method 3.  Use all the filter estimates and their associated raw
C                probabilities.

      ELSE IF (ISLCT.EQ.3) THEN
        CALL ACALC(ZXHP,PRBDUM,PORDER,NFLTR)

C444444444444444444444444444444444444444444444444444444444444444444444
C --- Method 4.  Use the two highest filter probabilities with their
C                raw probabilities.

      ELSE IF (ISLCT.EQ.4) THEN
        CALL ACALC(ZXHP,PRBDUM,PORDER,5)

C555555555555555555555555555555555555555555555555555555555555555555555
C --- Method 5.  Use the two highest filter probabilities and readjust
C                their probabilities so that the two equal one.

      ELSE IF (ISLCT.EQ.5) THEN
        TEMP=PRBNEW(PORDER(4,Bank),Bank)+PRBNEW(PORDER(5,Bank),Bank)
        PDUMMY(PORDER(4,Bank),Bank)=PRBNEW(PORDER(4,Bank),Bank)/TEMP
        PDUMMY(PORDER(5,Bank),Bank)=PRBNEW(PORDER(5,Bank),Bank)/TEMP
        CALL ACALC(ZXHP,PDUMMY,PORDER,5)

C?????????????????????????????????????????????????????????????????????
```

```
C --- If ISLCT is anything besides 1 to 5, then the default will be
C     Method 3.

      ELSE
        CALL ACALC(ZXHP,PRBDUM,PORDER,NFLTR)

      END IF

      RETURN

C                             END ADPCON
C=====================================================================
C=====================================================================
      END
```

```
      SUBROUTINE PULLOUT
C***********************************************************************
C --- Pull out arrays from common block rawdat which remain constant   *
C     among the controllers.                                           *
C     Recall MTXC32 is a 3-d vector to 2-d vector conversion routine.  *
C***********************************************************************

      INCLUDE 'DECLARR.TXT'

C . . . . . . . . . . . . . . . . . . . .
C  l o c a l   v a r i a b l e s
C . . . . . . . . . . . . . . . . . . . .

C_______________________________________________________________________

C*******************************************************************
C --- Pull out the A, B, R, H, and CQDCNT matrices from
C     the raw data.
C*******************************************************************
C      WRITE(*,*) 'START PULLOUT'
      CALL MTXC32(ZA,A,8,8,20,MODELN,1,0)
      CALL MTXC32(ZB,B,8,6,20,MODELN,1,0)
C     CALL MTXC32(ZC,C,29,29,20,MODELN,1,0)
      CALL MTXC32(ZR,R,7,7,20,MODELN,1,0)
C      WRITE(*,*)'A',A
C
      CALL MTXC32(ZH,H,7,14,20,MODELN,1,0)
      CALL MTXC32(ZCQDCN,CQDCNT,8,8,20,MODELN,1,0)
C*******************************************************************
C --- Create a 7 X 29 HT from the 7 X 14 H matrix which is in the chosen
C     truth model file.  This is done to retain matrix size integrity.
C     Where,
C
C     HT =
C
C H(1,1),...,H(1,8),H(1,9),0,0,0,H(1,10),0,0,0,H(1,11),0,0,0,...,H(1,14)
C H(2,1),...,H(2,8),H(2,9),0,0,0,H(2,10),0,0,0,H(2,11),0,0,0,...,H(2,14)
C    .     .      .        .  . . .    .     . . .    .    . . . .     .
C    .     .      .        .  . . .    .     . . .    .    . . . .     .
C    .     .      .        .  . . .    .     . . .    .    . . . .     .
C H(7,1) ...,H(7,8),H(7,9),0,0,0,H(7,10),0,0,0,H(7,11),0,0,0,...,H(7,14)
C
C*******************************************************************

      DO 1031 IA=1,29

        DO 1030 IB=1,7

          IF (IA.LE.8) THEN
            HT(IB,IA)=H(IB,IA)
          ELSEIF ((IA.EQ.9).OR.(IA.EQ.13).OR.(IA.EQ.17).OR.
     &            (IA.EQ.21).OR.(IA.EQ.25).OR.(IA.EQ.29)) THEN
            HT(IB,IA)=H(IB,INT(9+(IA-9)/4))
          ELSE
            HT(IB,IA)=0.
          END IF

 1030   CONTINUE

 1031 CONTINUE

C      WRITE(*,*)'END PULLOUT'
      RETURN

C               END PULLOUT
```

```
      SUBROUTINE EOM(T,X,DX)
C*****************************************************************
C  REVISION BLOCK 12 JULY -  1)common block CONTROLSZ modified *
C  to include array AUNEW - 2) subroutine name change to        *
C  1STACT - A1STACT; 2NDACT - A2NDACT; 4THACT - A4THACT - 3)    *
C  variables passed within A1STACT,A2NDACT,A4THACT  had names   *
C  modified to prevent an access error when changing a value    *
C  within a passing string that is a constant in the main       *
C  routine (ex: CALL A1STACT(2,1,60.0,-60.0,2.0) in main        *
C  routine - SUBROUTINE A1STACT(IX,IY,UPRTLIM,LWRTLIM,XBIAS)    *
C  one cannot change the value of any of these variables        *
C  within the string since the variable value will not match    *
C  hardwired value in the original code.  Yet we need to        *
C  convert these values from deg and deg/sec to rad and         *
C  rad/sec in the subroutine.) Fixed by changing names within   *
C  the subroutine argument list; UPRTLIM -> AUPRTLIM  etc       *
C                                                               *
C                                                               *
C                                                               *
C*****************************************************************
C                                                               *
C  SUBROUTINE EOM calculates the incremental dynamics of the    *
C  air vehicle by passing a set of ordinary differential        *
C  equations to SUBROUTINE DEABM, a differential equation       *
C  solving routine.  A 29 x 29 [A] matrix contains the data     *
C  describing the coefficients of the differential equations    *
C  to be solved simultaneously by DEABM.  SUBROUTINE EOM        *
C  calculates the effects from control surface deflections      *
C  and includes those incremental effects into the incremental *
C  dynamic effects for a single sample period.  The routine     *
C  includes actuator dynamics and position and rate limiting.   *
C  The true fourth order actuator dynamics models are           *
C  represented as first order within the system.                *
C                                                               *
C  Creation Date:  26 June 1991                                 *
C  Revision Date:  27 Nov  1991                                 *
C  Owner:  USAF/ASD/AFIT/EN                                     *
C                                                               *
C*****************************************************************

      INCLUDE 'DECLARR.TXT'

C . . . . . . . . . . . . . . . . . . .
C l o c a l   v a r i a b l e s
C . . . . . . . . . . . . . . . . . . .

      INTEGER I,J
      REAL BNL(8,6),X(29),DX(29),T

      REAL TPA1,TPA2,TPB2,TPC2,TPA4,TPB4,TPC4,TPD4,TPE4,XBIAS
      COMMON/ACTVAL1/TPA1
      COMMON/ACTVAL2/TPA2,TPB2,TPC2
      COMMON/ACTVAL4/TPA4,TPB4,TPC4,TPD4,TPE4


      DATA WFLAG,IACTORDR/0,1/
      DATA TPA1/20.2/
      DATA TPA2,TPB2,TPC2/1.0,1.0,1.0/
      DATA TPA4,TPB4,TPC4,TPD4,TPE4/1.492537E+07,268.67,2.53731E+04,
     &1.1492537E+06,1.492537E+07/


C----------------------------------------------------------------C
C                                                                C
C     M A I N   P R O C E S S I N G   A L G O R I T H M          C
```

```
C                                                                    C
C____________________________________________________________________C
C--------------------------------------------------------------------
C  D E F I N I T I O N   O F   T H E   S T A T E   V A R I A B L E S
C  ------------------------------------------------------------------
C
C  X(1) - THETA           X(9)  - LEFT STABILATOR POSITION
C  X(2) - VELOCITY        X(10) - LEFT STABILATOR RATE
C  X(3) - ALFA            X(11) - LEFT STABILATOR 3RD STATE
C  X(4) - PITCHRATE       X(12) - LEFT STABILATOR 4TH STATE
C  X(5) - ROLL ANGLE
C  X(6) - BETA            X(13) - RIGHT STABILATOR POSITION
C  X(7) - ROLL RATE       X(14) - RIGHT STABILATOR RATE
C  X(8) - PITCH RATE      X(15) - RIGHT STABILATOR 3RD STATE
C                         X(16) - RIGHT STABILATOR 4TH STATE

C  X(17) - LEFT FLAPERON POSITION
C  X(18) - LEFT FLAPERON RATE
C  X(19) - LEFT STABILATOR 3RD STATE
C  X(20) - LEFT STABILATOR 4TH STATE

C  X(21) - RIGHT FLAPERON POSITION
C  X(22) - RIGHT FLAPERON RATE
C  X(23) - RIGHT FLAPERON 3RD STATE
C  X(24) - RIGHT FLAPERON 4TH STATE

C  X(25) - RUDDER POSITION
C  X(26) - RUDDER RATE
C  X(27) - RUDDER 3RD STATE
C  X(28) - RUDDER 4TH STATE

C  X(29) - LEADING EDGE FLAP POSITION

C  -------------------------------------------------------
C        E N D   O F   S T A T E   D E F I N I T I O N S
C  -------------------------------------------------------


C
C   INITIALIZE  [B] MATRIX
C


      DO I = 1,8
        DO J = 1,6

          BNL(I,J) = B(I,J)

        ENDDO
      ENDDO
C      WRITE(*,*)'EOM SUBROUTINE '
C     WRITE(*,*)'B MATRIX = ',B

C      DO I=1,4
C       WRITE(*,*)('A(',I,',',J,') ',A(I,J),J=1,4)
C      ENDDO
C
C      DO I=1,8
C       WRITE(*,*)('BNL',I,',',J,') ',BNL(I,J),J=1,6)
C      ENDDO


C --- Allow 29 - state truth model to incorporate actuator dynamics
```

```
C     or the Dryden wind model (future expansion) or both.

C --- AIRCRAFT PERTURBATION STATES

      DX(1) = A(1,1)*X(1) + A(1,2)*X(2) + A(1,3)*X(3) + A(1,4)*X(4)
      DX(2) = A(2,1)*X(1) + A(2,2)*X(2) + A(2,3)*X(3) + A(2,4)*X(4)
      DX(3) = A(3,1)*X(1) + A(3,2)*X(2) + A(3,3)*X(3) + A(3,4)*X(4)
      DX(4) = A(4,1)*X(1) + A(4,2)*X(2) + A(4,3)*X(3) + A(4,4)*X(4)
      DX(5) = A(5,5)*X(5) + A(5,6)*X(6) + A(5,7)*X(7) + A(5,8)*X(8)
      DX(6) = A(6,5)*X(5) + A(6,6)*X(6) + A(6,7)*X(7) + A(6,8)*X(8)
      DX(7) = A(7,5)*X(5) + A(7,6)*X(6) + A(7,7)*X(7) + A(7,8)*X(8)
      DX(8) = A(8,5)*X(5) + A(8,6)*X(6) + A(8,7)*X(7) + A(8,8)*X(8)

C --- ACTUATOR STATES  (initialized to zero)

C ------  DIFFERENTIAL LEFT STABILATOR ----- (4TH ORDER ACTUATOR)
      DX(9) = 0.
      DX(10) = 0.
      DX(11) = 0.
      DX(12) = 0.

C ------  DIFFERENTIAL RIGHT STABILATOR ---- (4TH ORDER ACTUATOR)
      DX(13) = 0.
      DX(14) = 0.
      DX(15) = 0.
      DX(16) = 0.

C ------  LEFT FLAPERON -------------------- (4TH ORDER ACTUATOR)
      DX(17) = 0.
      DX(18) = 0.
      DX(19) = 0.
      DX(20) = 0.

C ------  RIGHT FLAPERON ------------------- (4TH ORDER ACTUATOR)
      DX(21) = 0.
      DX(22) = 0.
      DX(23) = 0.
      DX(24) = 0.

C ------  RUDDER  -------------------------- (4TH ORDER ACTUATOR)
      DX(25) = 0.
      DX(26) = 0.
      DX(27) = 0.
      DX(28) = 0.

C ------  LEADING EDGE FLAP ---------------- (1ST ORDER ACTUATOR)
      DX(29) = 0.


C  INCREMENTS TO THE EQUATIONS OF MOTION DUE TO ACTUATOR EFFECTS

      DX(2) = DX(2) + BNL(2,1)*X(9) + BNL(2,2)*X(13) + BNL(2,3)*X(17)
     &+ BNL(2,4)*X(21) + BNL(2,6)*X(29)

      DX(3) = DX(3) + BNL(3,1)*X(9) + BNL(3,2)*X(13) + BNL(3,3)*X(17)
     &+ BNL(3,4)*X(21) + BNL(3,6)*X(29)

      DX(4) = DX(4) + BNL(4,1)*X(9) + BNL(4,2)*X(13) + BNL(4,3)*X(17)
     &+ BNL(4,4)*X(21) + BNL(4,6)*X(29)

      DX(6) = DX(6) + BNL(6,1)*X(9) + BNL(6,2)*X(13) + BNL(6,3)*X(17)
     &+ BNL(6,4)*X(21) + BNL(6,5)*X(25)

      DX(7) = DX(7) + BNL(7,1)*X(9) + BNL(7,2)*X(13) + BNL(7,3)*X(17)
     &+ BNL(7,4)*X(21) + BNL(7,5)*X(25)
```

```
      DX(8) = DX(8) + BNL(8,1)*X(9) + BNL(8,2)*X(13) + BNL(8,3)*X(17)
     &+ BNL(8,4)*X(21) + BNL(8,5)*X(25)

C  INCREMENTS TO THE EQUATIONS OF MOTION DUE TO ACTUATOR EFFECTS

C       DX(2) = DX(2) + BNL(2,1)*AUNEW(1) + BNL(2,2)*AUNEW(2)
C     & + BNL(2,3)*AUNEW(3) + BNL(2,4)*AUNEW(4) + BNL(2,6)*AUNEW(6)

C       DX(3) = DX(3) + BNL(3,1)*AUNEW(1) + BNL(3,2)*AUNEW(2)
C     & + BNL(3,3)*AUNEW(3) + BNL(3,4)*AUNEW(4) + BNL(3,6)*AUNEW(6)

C       DX(4) = DX(4) + BNL(4,1)*AUNEW(1) + BNL(4,2)*AUNEW(2)
C     & + BNL(4,3)*AUNEW(3) + BNL(4,4)*AUNEW(4) + BNL(4,6)*AUNEW(6)

C       DX(6) = DX(6) + BNL(6,1)*AUNEW(1) + BNL(6,2)*AUNEW(2)
C     & + BNL(6,3)*AUNEW(3) + BNL(6,4)*AUNEW(4) + BNL(6,5)*AUNEW(5)

C       DX(7) = DX(7) + BNL(7,1)*AUNEW(1) + BNL(7,2)*AUNEW(2)
C     & + BNL(7,3)*AUNEW(3) + BNL(7,4)*AUNEW(4) + BNL(7,5)*AUNEW(5)

C       DX(8) = DX(8) + BNL(8,1)*AUNEW(1) + BNL(8,2)*AUNEW(2)
C     & + BNL(8,3)*AUNEW(3) + BNL(8,4)*AUNEW(4) + BNL(8,5)*AUNEW(5)

C-----------------------------------------------------------------C
C                A C T U A T O R     E F F E C T S                C
C-----------------------------------------------------------------C

C                  --------------------------------
C                  -   complete actuator reponses  -
C                  --------------------------------

C  NOTE:  ********************************************************
C                                                                *
C  A1STACT, A2NDACT, A4THACT provide complete actuator packages  *
C  including rate limiting, the appropriate lags, and filters,   *
C  and position limiting.                                        *
C                                                                *
C  ***************************************************************


C -------------  Differential Stabilators -------------

C      ---- LEFT STABILATOR ----- DX(9) - DX(12)

C      The flag IXVALZ controls which elements within the DX and
C      X array are utilized for each control surface

       IF(IACTORDR.EQ.1)CALL A1STACT(X,DX,9,1,60.0,-60.0,
     &21.0,-21.0,2.0)
       IF(IACTORDR.EQ.2)CALL A2NDACT(X,DX,9,1,60.0,-60.0,
     &21.0,-21.0,2.0)
       IF(IACTORDR.EQ.4)CALL A4THACT(X,DX,9,1,60.0,-60.0,
     &21.0,-21.0,2.0)

C      ---- RIGHT STABILATOR ---- DX(13) - DX(16)

       IF(IACTORDR.EQ.1)CALL A1STACT(X,DX,13,2,60.0,-60.0,
     &21.0,-21.0,2.0)
       IF(IACTORDR.EQ.2)CALL A2NDACT(X,DX,13,2,60.0,-60.0,
     &21.0,-21.0,2.0)
       IF(IACTORDR.EQ.4)CALL A4THACT(X,DX,13,2,60.0,-60.0,
```

```
     &21.0,-21.0,2.0)

C -----------------  Leading Edge Flap  ------ DX(29)

      CALL A1STACT(X,DX,29,6,30.0,-30.0,25.0,-2.0,0.0)

C ---------------  Differential Flaperons  -------------

C      ---- LEFT FLAPERON ----  DX(17) - DX(20)

      IF(IACTORDR.EQ.1)CALL A1STACT(X,DX,17,3,61.0,-61.0,
     &20.0,-23.0,1.5)
      IF(IACTORDR.EQ.2)CALL A2NDACT(X,DX,17,3,61.0,-61.0,
     &20.0,-23.0,1.5)
      IF(IACTORDR.EQ.4)CALL A4THACT(X,DX,17,3,61.0,-61.0,
     &20.0,-23.0,1.5)

C      ---- RIGHT FLAPERON ----  DX(21) - DX(24)

      IF(IACTORDR.EQ.1)CALL A1STACT(X,DX,21,4,61.0,-61.0,
     &20.0,-23.0,1.5)
      IF(IACTORDR.EQ.2)CALL A2NDACT(X,DX,21,4,61.0,-61.0,
     &20.0,-23.0,1.5)
      IF(IACTORDR.EQ.4)CALL A4THACT(X,DX,21,4,61.0,-61.0,
     &20.0,-23.0,1.5)


C -----------------------  Rudder ---------- DX(25) - DX(28)

      IF(IACTORDR.EQ.1)CALL A1STACT(X,DX,25,5,120.0,-120.0,
     &30.0,-30.0,0.0)
      IF(IACTORDR.EQ.2)CALL A2NDACT(X,DX,25,5,120.0,-120.0,
     &30.0,-30.0,0.0)
      IF(IACTORDR.EQ.4)CALL A4THACT(X,DX,25,5,120.0,-120.0,
     &30.0,-30.0,0.0)

C      WRITE(*,*)' AUNEW = ',AUNEW
C      WRITE(*,*)' END EOM'
      RETURN

C                    END EOM
C*************************************************************
C*************************************************************
      END
C
C -----------------------------------------------------------C
C                                                            C
C                   SUBROUTINE   SECTION                     C
C                                                            C
C -----------------------------------------------------------C
C
C
C
C  --------------------------------------
C     P R O C E S S   S U B R O U T I N E S
C  --------------------------------------
C
C
      SUBROUTINE A1STACT(X,DX,IXVALZ,IXAUNEW,AUPRTLIM,
     &ALWRTLIM,AUPPOSLM,ALWPOSLM,AXBIAS)
C
C  ******************************************************
C  *                                                    *
C  *  SUBROUTINE A1STACT calculates a 1st order actuator *
```

```
C  *  response.  This routine constructs a first order      *
C  *  state space model using the ONESS routine, rate       *
C  *  limit the signal, subtract a mechanical bias and      *
C  *  position limit the surface command.                   *
C  *                                                        *
C  *                                                        *
C  *                                                        *
C  *  Creation date: 27 June 1991                           *
C  *  Revision date: 27 Nov  1991                           *
C  *  Owner: USAF/ASD/AFIT/EN                               *
C  *                                                        *
C  ********************************************************

      INCLUDE 'DECLARR.TXT'

      REAL TPA1,UPRTLIM,LWRTLIM,UPPOSLIM,LWPOSLIM,XBIAS
      REAL DX(29),X(29)
      COMMON/ACTVAL1/TPA1

      CALL ONESS(X,DX,IXVALZ,IXAUNEW)

C
C     CONVERSION ROUTINE - convert bias, rate and position
C     limits to rad and rad/sec since the signal is in rad and
C     rad/sec
C
      DTOR = 3.1415926/180.0
      XBIAS = AXBIAS*DTOR
      UPRTLIM = AUPRTLIM*DTOR
      LWRTLIM = ALWRTLIM*DTOR
      UPPOSLIM = AUPPOSLM*DTOR
      LWPOSLIM = ALWPOSLM*DTOR


C     subtract mechanical bias terms

C     X(IXVALZ) = X(IXVALZ) - XBIAS

C     rate and position limit the surface commands
C
C      write(*,*)' 1st order act '
C
      IF((X(IXVALZ+1).GE.UPRTLIM).AND.(DX(IXVALZ+1).GT.0.0))THEN
        DX(IXVALZ+1)=0.0
        write(*,*)' upper rate limited'
      ENDIF

      IF((X(IXVALZ+1).LE.LWRTLIM).AND.(DX(IXVALZ+1).LT.0.0))THEN
        DX(IXVALZ+1)=0.0
        WRITE(*,*)' lower rate limited'
      ENDIF
C
      IF((X(IXVALZ).GE.UPPOSLIM).AND.(DX(IXVALZ).GT.0.0))THEN
        DX(IXVALZ)=0.0
        WRITE(*,*)' upper position limited'
      ENDIF

      IF((X(IXVALZ).LE.LWPOSLIM).AND.(DX(IXVALZ).LT.0.0))THEN
        DX(IXVALZ)=0.0
        WRITE(*,*)' lower position limited'
      ENDIF
C
      RETURN
      END
C
C ----------------------------------------------------
```

```
C
      SUBROUTINE A2NDACT(X,DX,IXVALZ,IXAUNEW,AUPRTLIM,
     &ALWRTLIM,AUPPOSLM,ALWPOSLM,AXBIAS)

C   ******************************************************
C   *                                                    *
C   *  SUBROUTINE A2NDACT calculates a 2nd order actuator *
C   *  response.  This routine constructs a second order *
C   *  state space model using the TWOSS routine, rate    *
C   *  limit the signal, subtract a mechanical bias and   *
C   *  position limit the surface command.                *
C   *                                                    *
C   *  Creation date:  27 June 1991                      *
C   *  Revision date:  11 July 1991                      *
C   *  Owner:  USAF/ASD/AFIT/EN                          *
C   *                                                    *
C   ******************************************************

      INCLUDE 'DECLARR.TXT'

      REAL TPA2,TPB2,TPC2,UPRTLIM,LWRTLIM,UPPOSLIM,LWPOSLIM,XBIAS
      REAL DX(29),X(29)
      COMMON/ACTVAL2/TPA2,TPB2,TPC2

      CALL TWOSS(X,DX,IXVALZ,IXAUNEW)

C
C     CONVERSION ROUTINE - convert bias, rate and position
C     limits to rad and rad/sec since the signal is in rad and
C     rad/sec
C
      DTOR = 3.1415926/180.0
      XBIAS = AXBIAS*DTOR
      UPRTLIM = AUPRTLIM*DTOR
      LWRTLIM = ALWRTLIM*DTOR
      UPPOSLIM = AUPPOSLM*DTOR
      LWPOSLIM = ALWPOSLM*DTOR


C     subtract mechanical bias terms

C     X(IXVALZ) = X(IXVALZ) - XBIAS


C     rate and position limit the surface commands


      IF((X(IXVALZ+1).GE.UPRTLIM).AND.(DX(IXVALZ+1).GT.0.0))THEN
        DX(IXVALZ+1)=0.0
      ENDIF
      IF((X(IXVALZ+1).LE.LWRTLIM).AND.(DX(IXVALZ+1).LT.0.0))THEN
        DX(IXVALZ+1)=0.0
      ENDIF
      IF((X(IXVALZ).GE.UPPOSLIM).AND.(DX(IXVALZ).GT.0.0))THEN
        DX(IXVALZ)=0.0
      ENDIF
      IF((X(IXVALZ).LE.LWPOSLIM).AND.(DX(IXVALZ).LT.0.0))THEN
        DX(IXVALZ)=0.0
      ENDIF

      RETURN
      END
C
C -------------------------------------------------------
C
      SUBROUTINE A4THACT(X,DX,IXVALZ,IXAUNEW,AUPRTLIM,
```

```
     &ALWRTLIM,AUPPOSLM,ALWPOSLM,AXBIAS)
C  ****************************************************
C  *                                                  *
C  *  SUBROUTINE A4THACT calculates a 4th order actuator *
C  *  response.  This routine constructs a fourth order  *
C  *  state space model using the FOURSS routine, rate   *
C  *  limit the signal, subtract a mechanical bias and   *
C  *  position limit the surface command.                *
C  *                                                  *
C  *  Creation date:  27 June 1991                    *
C  *  Revision date:  11 July 1991                    *
C  *  Owner:  USAF/ASD/AFIT/EN                        *
C  *                                                  *
C  ****************************************************

      INCLUDE 'DECLARR.TXT'

      INTEGER IXVALZ,IXAUNEW
      REAL TPA4,TPB4,TPC4,TPD4,TPE4,UPRTLIM,LWRTLIM,UPPOSLIM,
     &LWPOSLIM,XBIAS,DTOR
      REAL DX(29),X(29)
      COMMON/ACTVAL4/TPA4,TPB4,TPC4,TPD4,TPE4

C     WRITE(*,*)'DX(IXVALZ):',DX(IXVALZ),DX(IXVALZ+1)
      CALL FOURSS(X,DX,IXVALZ,IXAUNEW)
C     WRITE(*,*)'DX(IXVALZ):',DX(IXVALZ),DX(IXVALZ+1)
C
C     CONVERSION ROUTINE - convert bias, rate and position
C     limits to rad and rad/sec since the signal is in rad and
C     rad/sec
C
      DTOR = 3.1415926/180.0
      XBIAS = AXBIAS*DTOR
      UPRTLIM = AUPRTLIM*DTOR
      LWRTLIM = ALWRTLIM*DTOR
      UPPOSLIM = AUPPOSLM*DTOR
      LWPOSLIM = ALWPOSLM*DTOR

C     WRITE(*,*)UPPOSLIM,UPRTLIM,LWPOSLIM,LWRTLIM

C     subtract mechanical bias terms

C     X(IXVALZ) = X(IXVALZ) - XBIAS

C     rate and position limit the surface commands


C      WRITE(*,*)'X(IXVALZ),UPPER POS LIMIT',X(IXVALZ),UPPOSLIM
C      WRITE(*,*)'DX(IXVALZ) =',DX(IXVALZ)
C      WRITE(*,*)'X(IXVALZ),LOWER POS LIMIT',X(IXVALZ),LWPOSLIM
C      WRITE(*,*)'DX(IXVALZ) =',DX(IXVALZ)
C      WRITE(*,*)'X(IXVALZ+1),UPPER RATE LIMIT',X(IXVALZ+1),UPRTLIM
C      WRITE(*,*)'DX(IXVALZ+1) =',DX(IXVALZ+1)
C      WRITE(*,*)'X(IXVALZ+1),LOWER RATE LIMIT',X(IXVALZ+1),LWRTLIM
C      WRITE(*,*)'DX(IXVALZ+1) =',DX(IXVALZ+1)

      IF(IXVALZ.EQ.9)THEN
C      WRITE(*,*)'X(9-10)',X(9),X(10)
      ENDIF

      IF(IXVALZ.EQ.13)THEN
C      WRITE(*,*)'X(13-14)',X(13),X(14)
      ENDIF

      IF((X(IXVALZ+1).GE.UPRTLIM).AND.(DX(IXVALZ+1).GT.0.0))THEN
```

```
          DX(IXVALZ+1)=0.0
        ENDIF
        IF((X(IXVALZ+1).LE.LWRTLIM).AND.(DX(IXVALZ+1).LT.0.0))THEN
          DX(IXVALZ+1)=0.0
        ENDIF
        IF((X(IXVALZ).GE.UPPOSLIM).AND.(DX(IXVALZ).GT.0.0))THEN
          DX(IXVALZ)=0.0
        ENDIF
        IF((X(IXVALZ).LE.LWPOSLIM).AND.(DX(IXVALZ).LT.0.0))THEN
          DX(IXVALZ)=0.0
        ENDIF

        IF(X(IXVALZ+1).GE.UPRTLIM.AND.DX(IXVALZ+1).GT.0.0)THEN
          WRITE(*,*)'UPPER RATE LIM , ACT = :',IXVALZ
        ENDIF
        IF(X(IXVALZ+1).LE.LWRTLIM.AND.DX(IXVALZ+1).LT.0.0)THEN
          WRITE(*,*)'LOWER RATE LIM , ACT = :',IXVALZ
        ENDIF
        IF(X(IXVALZ).GE.UPPOSLIM.AND.DX(IXVALZ).GT.0.0)THEN
          WRITE(*,*)'UPPER POS LIM , ACT = :',IXVALZ
        ENDIF

        IF(X(IXVALZ).LE.LWPOSLIM.AND.DX(IXVALZ).LT.0.0)THEN
          WRITE(*,*)'LOWER POS LIM , ACT = :',IXVALZ
        ENDIF

C       WRITE(*,*)' PCSITION LIMIT - DX(IXVALZ) ',DX(IXVALZ)
C       WRITE(*,*)' RATE LIMIT - DX(IXVALZ) ',DX(IXVALZ+1)
        RETURN
        END
C
C
C
C   --------------------------------------------------
C      F U N C T I O N A L   S U B R O U T I N E S
C   --------------------------------------------------
C
C
        SUBROUTINE ONESS(X,DX,IXVALZ,IXAUNEW)
C  ***************************************************
C  *                                                 *
C  *  SUBROUTINE ONESS provides a first order        *
C  *  state space model representing a first order   *
C  *  actuator                                       *
C  *                                                 *
C  *  Form:  O(t)        TPA                         *
C  *         ----  =  --------                       *
C  *         I(t)     S + TPA                        *
C  *                                                 *
C  *                                                 *
C  *  Creation date: 28 June 1991                    *
C  *  Revision date: 11 July 1991                    *
C  *  Owner:  USAF/ASD/AFIT/EN                       *
C  *                                                 *
C  ***************************************************

        INCLUDE 'DECLARR.TXT'

        REAL TPA1
        REAL DX(29),X(29)

        COMMON/ACTVAL1/TPA1

C hardwire actuator lag coefficient for 1st order modeling
C of higher order actuators except the leading edge flap
```

```
      IF(IXVALZ.NE.29)THEN
        TPA1 = 20.2
      ELSE
        TPA1 = 16.0
      ENDIF


      DX(IXVALZ) = -TPA1*X(IXVALZ) + TPA1*AUNEW(IXAUNEW)

      IF(IXVALZ.NE.29)THEN
        DX(IXVALZ + 1) = 0.0
        DX(IXVALZ + 2) = 0.0
        DX(IXVALZ + 3) = 0.0
      ENDIF

      RETURN
      END
C
C ----------------------------------------------------------
C
      SUBROUTINE TWOSS(X,DX,IXVALZ,IXAUNEW)
C ****************************************************
C *                                                  *
C *   SUBROUTINE TWOSS provides a second order       *
C *   state space model representing a second order  *
C *   actuator                                       *
C *                                                  *
C *   Form:  O(t)          TPA                       *
C *          ----  =  --------------                 *
C *          I(t)     S^2 +TPBS +TPC                 *
C *                                                  *
C *                                                  *
C *   Creation date: 28 June 1991                    *
C *   Revision date: 11 July 1991                    *
C *   Owner:  USAF/ASD/AFIT/EN                       *
C *                                                  *
C ****************************************************

      INCLUDE 'DECLARR.TXT'

      REAL TPA2,TPB2,TPC2
      REAL DX(29),X(29)

      COMMON/ACTVAL2/TPA2,TPB2,TPC2

      DX(IXVALZ) = X(IXVALZ + 1)
      DX(IXVALZ + 1) = -TPB2*X(IXVALZ+1) - TPC2*X(IXVALZ)
     &+ TPA2*AUNEW(IXANEW)
      DX(IXVALZ + 2) = 0.0
      DX(IXVALZ + 3) = 0.0

      RETURN
      END
C
C ----------------------------------------------------------
C
      SUBROUTINE FOURSS(X,DX,IXVALZ,IXAUNEW)
C ****************************************************
C *                                                  *
C *   SUBROUTINE FOURSS provides a fourth order      *
C *   state space model representing a fourth order  *
C *   actuator                                       *
C *                                                  *
C *   Form:  O(t)                 TPA                *
C *          ----  =  ------------------------------ *
C *          I(t)     S^4 +TPBS^3 +TPCS^2 + TPDS + TPE *
```

```
C  *                                                         *
C  *                                                         *
C  *  Creation date: 28 June 1991                            *
C  *  Revision date: 11 July 1991                            *
C  *  Owner:  USAF/ASD/AFIT/EN                               *
C  *                                                         *
C  ***********************************************************

      INCLUDE 'DECLARR.TXT'

      REAL TPA4,TPB4,TPC4,TPD4,TPE4
      REAL DX(29),X(29)

      COMMON/ACTVAL4/TPA4,TPB4,TPC4,TPD4,TPE4
C      WRITE(*,*)'ACTUATOR CONSTANTS'
C      WRITE(*,*)TPA4,TPB4,TPC4,TPD4,TPE4

      DX(IXVALZ) = X(IXVALZ + 1)
      DX(IXVALZ + 1) = X(IXVALZ + 2)
      DX(IXVALZ + 2) = X(IXVALZ + 3)
      DX(IXVALZ + 3) = -TPE4*X(IXVALZ) - TPD4*X(IXVALZ + 1)
     &-TPC4*X(IXVALZ + 2) - TPB4*X(IXVALZ + 3) + TPA4*AUNEW(IXAUNEW)

      RETURN
      END
```

```
      SUBROUTINE KFILT(T,X,DX)
C*********************************************************************
C     Subroutine to incorporate the Kalman filter into the loop.
C*********************************************************************

      INCLUDE 'DECLARR.TXT'

C . . . . . . . . . . . . . . . . . . .
C  l o c a l   v a r i a b l e s
C . . . . . . . . . . . . . . . . . . .

      REAL AWORK(14,1),BWORK(14,1)
      REAL XHP(14),XHM(14),GKF(14,7),X(29),DX(29)
      REAL PRBOLD(20,15),PRBTMP(20,15),ZXHM(14,20)
      REAL PHIX(14,14),BD(14,6),ZXHP(14,20)
      INTEGER  I,J,K,IFLTR

      SAVE ZXHM,PRBOLD,PRBTMP,ZXHP

C*********************************************************************
C --- Set PRBOLD to the last probabilities from last time through and
C     zero out PRBTMP.
C*********************************************************************

      IF ((Bankflag.eq.1) .and. (initv2.eq.0)) THEN
        DO 1781 ti=4,nfltr(bank)
          PRBOLD(ti,bank)=PRBNEW(ti,bank)
          PRBTMP(ti,bank)=0.0
 1781   Continue
        Initv2=1
      END IF

C*********************************************************************
C --- If this is the first call to KFILT after the start of a time
C     response loop, inidicated with INITV=1, initialize variables.
C     Do to 20 for hierarchical modeling.
C*********************************************************************

      IF (INITV.EQ.1) THEN
        DO 20 I=1,20

          DO 10 J=1,14
            ZXHP(J,I)=0.
            ZXHM(J,I)=0.
   10     CONTINUE

          IF ((I.ne.modeln1) .and. (I.ne.modeln2)
     +                       .and. (I.ne.modeln3)) THEN
            PRBOLD(I,Bank)=(1.-PRBFLTRT0)/FLOAT(NFLTR(Bank)-4)
            IF (I.eq.4) PRBOLD(I,Bank)=PRBFLTRT0
            PRBTMP(I,Bank)=0.
          END IF

   20   CONTINUE
        INITV=0
      END IF

C*********************************************************************
C --- Calculate new Xhat+ (XHP) and un-normalized probabilties (PRBTMP)
C     (i.e., numerator of equation 10-104 of Maybeck -- for our implementation
C     we can either strip off the beta term or leave as is by setting the
C     BETAFLG to 0 or 1, respectively) by calls to UPDATE for each of the
C     NFLTR total elemental filters.
C
C --- Also calculate the weighted sum of Xhat+ (XHPSUM), which is done by
C     a call to ADPCON which is then passed to CONTROLLER to compute the
```

```
C     input to the actuators (AUNEW).
C*********************************************************************

      DO 100 IFLTR=1,NFLTR(Bank)
        IF ((IFLTR.ne.modeln1) .and. (IFLTR.ne.modeln2)
     +                         .and. (IFLTR.ne.modeln3)) THEN

          CALL MTXC21(ZXHM,XHM,14,20,IFLTR,1,0)
          CALL MTXC21(ZXHP,XHP,14,20,IFLTR,1,0)
          CALL MTXC32(ZGKF,GKF,14,7,20,IFLTR,bank,0)
          CALL UPDATE(XHM,XHP,GKF,PRBTMP,PRBOLD,IFLTR,T)
          CALL MTXC21(ZXHP,XHP,14,20,IFLTR,1,1)

        END IF
  100 CONTINUE

C*********************************************************************
C --- Calculate the weighted sum Xhat+ (XHPSUM).
C*********************************************************************

      CALL ADPCON(PRBTMP,PRBOLD,ZXHP)

C*********************************************************************
C --- Calculate the control vector (AUNEW).
C*********************************************************************

      CALL CNTRL(T,X,DX)

C*********************************************************************
C --- Propagate estimate forward in time (create a new Xhat-)
C     but do not propagate it for the actual aircraft
C     configuation (indicated by modeln1, modeln2, and modeln3.)
C*********************************************************************

      DO 200 K=1,NFLTR(Bank)
        IF ((K.ne.modeln1) .and. (K.ne.modeln2)
     +                     .and. (K.ne.modeln3)) THEN

          CALL MTXC21(ZXHP,XHP,14,20,K,1,0)
          CALL MTXC32(ZBD,BD,14,6,20,K,bank,0)
          CALL MTXC32(ZPHIX,PHIX,14,14,20,K,bank,0)
          CALL MATML(PHIX,XHP,AWORK,14,14,1)
          CALL MATML(BD,AUNEW,BWORK,14,6,1)
          CALL MATAD(AWORK,BWORK,XHM,14,1)
          CALL MTXC21(ZXHM,XHM,14,20,K,1,1)

C ///////////////////////////////////////////////////////
C                    CODE CHECK
C ///////////////////////////////////////////////////////
C         IF((T.GE.3.0).AND.(T.LE.3.3))THEN
C          IF(K.EQ.4)THEN
C            WRITE(18,*)'KFILTER #',K,' AT TIME ',T
C            WRITE(18,*)'XHP',XHP
C            WRITE(18,*)'PHIX',PHIX
C            WRITE(18,*)'AWORK',AWORK
C            WRITE(18,*)'BD ',BD
C            WRITE(18,*)'AUNEW',AUNEW
C            WRITE(18,*)'BWORK',BWORK
C            WRITE(18,*)'XHM',XHM
C            WRITE(18,*)'ZXHM',ZXHM
C          ENDIF
C         END IF
        END IF
  200 CONTINUE

      RETURN


C                            END KFILT
C---------------------------------------------------------------------
C---------------------------------------------------------------------
      END
```

```
      SUBROUTINE UPDATE(XHM,XHP,GKF,PRBTMP,PRBOLD,IFLTR,T)
C*********************************************************************
C     Subroutine called from KFILT that updates the state estimates
C     using the latest measurement vector, it also calculates the
C     residuals, the individual scalar residuals, and the probability
C*********************************************************************

      INCLUDE 'DECLARR.TXT'

C . . . . . . . . . . . . . . . . . . . . .
C     l o c a l   v a r i a b l e s
C . . . . . . . . . . . . . . . . . . . . .

      INTEGER INDX,IFAIL,IFLTR,ITEMPZZ,ITEMPBAZ
      REAL CWORK(7,1),DWORK(7,1),EWORK(14,1),FWORK(1,7),GWORK(1,1)
      REAL XHM(14),XHP(14),GKF(14,7),RESID(7,1),TRESID(1,7)
      REAL PRBOLD(20,15),PRBTMP(20,15),AK(7,7),AKINV(7,7),TMP1
C
C     declaration for code checking variables
C
      REAL hfltdum(7,8),xfltdum(8,1),zfltstat(7,1)
      REAL hfltcon(7,6),xfltcon(6,1),zfltcon(7,1)

      double precision ssqqrt
      double precision sumrakr(7)
      double precision Ck,Ck1
      integer jk

C*********************************************************************
C       itime   = The current time.
C       rakr    = The residual vector transpose times the Ak
C                 inverse matrix times the residual vector.
C       jk      = Indicates which Monte Carlo iteration is
C                 in progress.
C       m       = Number of sensors.
C       sumrakr = The sum of rakr used in the Log Likelihood
C                 calculation.
C*********************************************************************

      m=7.0
      pi=3.141592654

C*********************************************************************
C --- Zero out sumrakr array.
C*********************************************************************

      Do 11 jk=1,7
        sumrakr(jk)=0.0
 11   Continue

C*********************************************************************
C --- Update Xhat- with latest measurement data.
C     so that Xhat+=Xhat- + K[z - HXhat-]
C*********************************************************************

      CALL MTXC32(ZH,H,7,14,20,IFLTR,bank,0)
      CALL MATML(H,XHM,CWORK,7,14,1)
C      CWORK(4,1)=0.0
C      CWORK(7,1)=0.0
* ///////////////////////////////////////////////////////////
*               CODE CHECK
* ///////////////////////////////////////////////////////////
C      IF((T.GE.3.0).AND.(T.LE.3.3))THEN
C        IF(IFLTR.EQ.4)THEN
C          WRITE(18,*)'FILTER #',IFLTR
C          WRITE(18,*)'H ',H
```

```
C          WRITE(18,*)'XHM',XHM
C          WRITE(18,*)'Z',Z
C         ENDIF
C       ENDIF
* ////////////////////////////////////////////////////
      CALL MATSB(Z,CWORK,DWORK,7,1)

      IF((T.GE.2.95).AND.(T.LE.3.3))THEN
        IF(IFLTR.EQ.5)THEN

          do ifltdum=1,7
            do jfltdum=1,8
            hfltdum(ifltdum,jfltdum)=h(ifltdum,jfltdum)
            enddo
          enddo

          do ifltdum = 1,8
            xfltdum(ifltdum,1)=xhm(ifltdum)
          enddo

          CALL MATML(hfltdum,xfltdum,zfltstat,7,8,1)

          write(19,*)'filter #5 state portion of h*xhat-'
          write(19,*)'time, h'
          write(19,*)t,hfltdum
          write(19,*)' x '
          write(19,*)xfltdum
          write(19,*)' z '
          write(19,*)zfltstat

C   controller portion broken out of h matrix for check

          do ifltdum=1,7
            do jfltdum=1,6
            hfltcon(ifltdum,jfltdum)=h(ifltdum,jfltdum + 8)
            enddo
          enddo

          do ifltdum = 1,6
            xfltcon(ifltdum,1)=xhm(ifltdum + 8)
          enddo

          CALL MATML(hfltcon,xfltcon,zfltcon,7,6,1)

          write(19,*)'filter #5 controller portion of h*xhat-'
          write(19,*)'time, h'
          write(19,*)t,hfltcon
          write(19,*)' x '
          write(19,*)xfltcon
          write(19,*)' z '
          write(19,*)zfltcon

C          WRITE(19,*)T,Z(7),CWORK(7,1),DWORK(7,1)
        ENDIF
      ENDIF

*/////////////////////////////////////////////////////*
*            Modification for bias calculation
*/////////////////////////////////////////////////////*
*              correction for bias in LH Stabilator
C
C       If((IFLTR.EQ.5).AND.(T.GT.3.0))THEN
C         DWORK(7,1)=DWORK(7,1) + 0.008
C       Endif
C
```

```
*////////////////////////////////////////////////////////////////*
      CALL MATML(GKP,DWORK,EWORK,14,7,1)
      CALL MATAD(XHM,EWORK,XHP,14,1)

C*********************************************************************
C --- Calculate the residuals
C
C --- Also calculate the single scalar residuals
C      R^T * (AK)^-1 * R
C      for only two of the ten Monte Carlo loops (1 and 7)
C      Stored in RAKR(time;r1,r2,r3,r4,r5,r6,r7;loop1 or 7;filter id)
C
C --- These single scalar residuals are not currently used for
C      decision making processes in the code.  They are here to
C      verify, correct identification convergence
C      of the currently implemented MMAESIM, which strips off the
C      beta term.
C*********************************************************************

C*********************************************************************
C --- Get the individual AK inverse out of storage.
C*********************************************************************

      CALL MTXC32(ZAKINV,AKINV,7,7,20,IFLTR,bank,0)
      CALL MTXC32(ZAK,AK,7,7,20,IFLTR,bank,0)

C*********************************************************************
C --- Calculate each of the seven residual vector components.
C*********************************************************************

      DO 10 INDX=1,7
        RESID(INDX,1)=Z(INDX)-CWORK(INDX,1)
        TRESID(1,INDX)=RESID(INDX,1)
        itempbaz=ifltr-3
        IF (IJK.EQ.1) THEN
          RSSAVE(itime,indx,itempbaz)=RESID(INDX,1)
          BUZSAVE(itime,indx,itempbaz)=2.0*SQRT(AK(INDX,INDX))
c          write(*,*)'-------   update  ------- ',itime
c          write(*,*)buzsave(itime,indx,itempbaz),indx,itempbaz
c          write(*,*)ak(indx,indx)
        END IF

C*********************************************************************
C --- Take single scalar residuals for loops 1 and 7
C*********************************************************************

        IF ((ijk.eq.1) .or. (ijk.eq.7)) THEN
          rakr(itime,indx,ijk,IFLTR)=(resid(indx,1)**2)*
     +                               akinv(indx,indx)
        END IF

 10   CONTINUE

C*********************************************************************
C --- Calculate the individual probability coefficient
C      (numerator of Eq'n 10-104 of Maybeck V.2.)
C*********************************************************************

      CALL MATML(TRESID,AKINV,FWORK,1,7,7)
      CALL MATML(FWORK,resid,GWORK,1,7,1)

C*********************************************************************
C --- Recall, ZDETAK contains the determinant of AK.
C      These were determined separately in Matrixx and read in
C      subroutine GETDAT.
```

```
C
C --- Although GWORK is defined as an array, it is really the scalar
C     term r^T * Akinv * r.  It was defined as an array since it is the
C     result of the matrix multiplications above.
C*********************************************************************

      TMP1=(-.5)*GWORK(1,1)
      IF (TMP1.LE.-50.) TMP1=(-50.)
      ssqqrt=sqrt( abs(zdetak(IFLTR,Bank)) )

C      write(*,*)'itime, ifltr: ',itime,ifltr
C      write(*,*)'ssqqrt: ',ssqqrt
C*********************************************************************
C --- Recall, if BETAFLG is 0, the beta term is stripped off
C             if BETAFLG is 1, the beta term remains
C     If for some reason BETAFLG is something else, the default
C     will strip off the beta term
C*********************************************************************
C      write(*,*)'pi, m: ',pi, m
      Ck1=ssqqrt*(2.0*pi)**(m/2.0)
C      write(*,*)'ck1: ',ck1
      Ck=log(ck1)

      IF (BETAFLG.ne.1) Ck1=1
C      write(*,*)'ck, ck1: ',ck,ck1
      PRBTMP(IFLTR,Bank)=(PRBOLD(IFLTR,Bank)*EXP(TMP1))/Ck1

C**********************************************************************
C --- Now determine whether sensor failure exists using the development
C     on pages 229-231 of Maybeck V.1.  Use Equation 5-67, setting
C     N=10.  Here Ck=Log(ck1) (Log=natural logarithm) where ck1 is
C     defined above.
C
C     First, sum the kth residual and kth diagonal term of Ak(tj)
C     from j=i-N+1 to i, where i is the current time index (ti, here
C     called 'itime').
C     Note, we have to skip the first ten (N=10) increments so
C     we can begin summing at time=0.
C     Here, 'indx' takes the place of 'k' in Eq'n 5-67.
C     Note, '7' in the indx loop, is the number of rows in the H matrix.
C     Do this for the first Monte Carlo iteration (ijk=1).
C*********************************************************************

C      IF (ijk.ne.1) go to 30

C      IF (itime.lt.10) go to 30

C       DO 20 jk=(itime-10+1),itime
C        DO 15 indx=1,7
C          sumrakr(indx)=sumrakr(indx) + rakr(jk,indx,ijk,IFLTR)
C          IF (sumrakr(indx).gt.1.e30) sumrakr(indx)=1.0E30
C          Lk(itime,indx,IFLTR)=Ck-0.5*sumrakr(indx)
C 15     CONTINUE
C 20   CONTINUE

C 30   CONTINUE

      RETURN

C                          END UPDATE
C=====================================================================
C=====================================================================
      END
```

```
      SUBROUTINE ACALC(ZXHP,P,PORDER,ICNT)

      INCLUDE 'DECLARR.TXT'

C . . . . . . . . . . . . . . . . . . . . . .
C  l o c a l   v a r i a b l e s
C . . . . . . . . . . . . . . . . . . . . . .

      INTEGER PORDER(20,15),ICNT(15),I,J

      REAL ZXHP(14,20),P(20,15)

C_____________________________________________________________________

C*********************************************************************
C --- Calculate the weighted sum of the state estimate, XHPSUM.
C     Here, the J loop loops through 8 times, which is the number of
C     plant states.  ICNT depends on what method of calculation is
C     to be used as determined in ADPCON.
C*********************************************************************

      DO 20 J=1,14
        XHPSUM(J)=0.
        DO 10 I=1,ICNT(Bank)
          IF ((I.ne.modeln1) .and. (I.ne.modeln2)
     +                       .and. (I.ne.modeln3)) THEN
            XHPSUM(J)=XHPSUM(J)+
     +                (ZXHP(J,PORDER(I,Bank))*P(PORDER(I,Bank),Bank))
            IF (XHPSUM(j).gt. 1.0e15) XHPSUM(j)= 1.0e15
            IF (XHPSUM(j).lt.-1.0e15) XHPSUM(j)=-1.0e15
          END IF
 10     CONTINUE
 20   CONTINUE

      RETURN

C                          END ACALC
C=====================================================================
C=====================================================================
      END
```

```
      SUBROUTINE COMMAND(FEC,FAC,FPC,T)
C ---- Provides the vista flight control system [CNTRL.FOR]
C       with the command signals for the longitudinal axis
C       [FEC], the lateral command [FRC], and the directional
C       command [FPC].  Additionally, simulation results have
C       indicated the need for a dither signal to "shake up"
C       the system and aid the filters in their identification
C       tasks.

      INCLUDE 'DECLARR.TXT'
      REAL FEC,FAC,FPC,DON,T,omega1,omega2,omega3

      SAVE

C     COMMAND SIGNALS


C . . . . . . . . . . . . . . . . . . . . . .
C  Dither signal generation
C    --NOTE:  if a control command is to be used,
C             it must be added to the below
C             created dither signal.
C . . . . . . . . . . . . . . . . . . . . . .
C
C
C    P U L S E D   D I T H E R   S I G N A L
C
C              NON - SUBLIMINAL
C . . . . . . . . . . . . . . . . . . . . . .

        Don=amod(t,3.0)

c         IF((Don.ge.0.0).and.(Don.lt.0.125))THEN
c           FEC = 13.5
c           FAC = -7.5
c           FPC = 24.0
c          ELSE IF((Don.ge.0.125).and.(Don.lt.0.25))THEN
c            FEC =-13.0
c            FAC = 7.5
c            FPC =-24.0
c          ELSE
c            FEC = 0.0
c            FAC = 0.0
c            FPC = 0.0
c         END IF
C
C . . . . . . . . . . . . . . . . . . . . . .
C
C
C    P U L S E D   D I T H E R   S I G N A L
C
C                   SUBLIMINAL
C . . . . . . . . . . . . . . . . . . . . . .
C

C       Don=amod(t,3.0)

          IF((Don.ge.0.0).and.(Don.lt.0.125))THEN
             FEC = 16.2
             FAC = -7.0
             FPC = 30.0
           ELSE IF((Don.ge.0.125).and.(Don.lt.0.25))THEN
             FEC =-17.5
             FAC =  7.5
             FPC = 30.0
```

```
          ELSE
            FEC = 0.0
            FAC = 0.0
            FPC = 0.0
         END IF

C
C . . . . . . . . . . . . . . . . . . . . . . . . . .
C
C    S I N U S O I D A L   D I T H E R   S I G N A L
C
C . . . . . . . . . . . . . . . . . . . . . . . . . .
C

C    FREQUENCY

        omega1 = 15.0
        omega2 = 15.0
        omega3 = 15.0

C    SIGNAL


c       If(t.ge.3.4)then

              FEC= 12.0*sin(omega1*t)
              IF(FEC.LT.0.0)FEC=12.5*sin(omega1*t)
              FAC= -11.0*sin(omega2*t)
              FPC= 30.0*sin(omega3*t)

c      else 18/20.5/-7/22
c             FEC=28.0*sin(omega1*t)
c             IF(FEC.LT.0.0)FEC=30.5*sin(omega1*t)
c             FAC= -7.0*sin(omega2*t)
c             FPC= 22.0*sin(omega3*t)
c       endif

C   . . . . . . . . . . . . . . . . . . . . . . . . .
C
C
C   U S E R   D E F I N E D   D I T H E R   S I G N A L
C
C
C   . . . . . . . . . . . . . . . . . . . . . . . . .

C      Don=amod(t,3.0)

C      IF((Don.ge.0.0).and.(Don.lt.0.1))THEN
C        FEC = 15.2
C        FAC = 16.0
C        FPC = 55.0
C      ELSE IF((Don.ge.0.1).and.(Don.lt.0.125))THEN
C        FEC = (-15.2/0.025)*(Don - 0.1) + 15.2
C        FAC = (-16.0/0.025)*(Don - 0.1) + 16.0
C        FPC = (-55.0/0.025)*(Don - 0.1) + 55.0
C      ELSE IF((Don.ge.0.125).and.(Don.lt.0.3))THEN
C        FEC =-16.5
C        FAC =-14.0
C        FPC =-50.0
C      ELSE IF((Don.ge.0.3).and.(Don.lt.0.325))THEN
C        FEC = (16.5/0.025)*(Don - 0.3) - 16.5
C        FAC = (14.0/0.025)*(Don - 0.3) - 14.0
C        FPC = (50.0/0.025)*(Don - 0.3) - 50.0
C      ELSE
C        FEC = 0.0
C        FAC = 0.0
```

```
C        FPC = 0.0
C      END IF

C  . . . . . . . . . . . . . . . . . . . . . . . . . . .
C
C
C  P I L O T   P U R P O S E F U L   C O M M A N D S
C
C
C  . . . . . . . . . . . . . . . . . . . . . . . . . . .

C     Pitch Path

C       IF((t.GT.2.95).and.(t.LT.3.15))THEN
C         FEC=13.5
C         FAC=20.0
C        FPC=-40.0
C       ENDIF
C        IF((T.GT.6.0).and.(t.LT.6.15))THEN
c          FEC=18.5
C          FAC=-15.0
C          FPC=20.
C        ELSE IF((T.GE.6.15).AND.(T.LT.6.3))THEN
C          FAC=15
C          FPC=-20.
C        ENDIF
C     Roll Path


C     Yaw Path


      RETURN
      END

      SUBROUTINE COPYMT(A,B,N,M)
C**********************************************************************
C THIS ROUTINE COPIES A REAL MATRIX A INTO REAL MATRIX B.
C BOTH MATRICES ARE OF DIMENSION N BY M.
C**********************************************************************
C
      INTEGER I,J,N,M
      REAL A(N,M),B(N,M)
      DO 100 I=1,N
         DO 100 J=1,M
            B(I,J)=A(I,J)
 100  CONTINUE
      RETURN
C/
C//////////////////////////////////////////////////////////////////////
C                             END COPYMT
C----------------------------------------------------------------------
      END
```

```
      SUBROUTINE DSORT(Values,Numval,INDEX)
C************************************************************************
C --- This subroutine sorts the probabilities from highest to lowest.
C     Here, Values = PRBNEW
C           Numval = NFLTR
C           INDEX  = PORDER
C************************************************************************

      INCLUDE 'DECLARR.TXT'

C . . . . . . . . . . . . . . . . . . . . . .
C  l o c a l   v a r i a b l e s
C . . . . . . . . . . . . . . . . . . . . . .

      INTEGER Numval(15),INDEX(20,15),ITMP,IC,J,K
      REAL Values(20,15),VTMP(20),TEMP

C_______________________________________________________________________


      DO 10 J=1,Numval(Bank)
        IF ((J.ne.modeln1) .and. (J.ne.modeln2)
     +                      .and. (J.ne.modeln3)) THEN
          VTMP(J)=Values(j,Bank)
          INDEX(J,Bank)=J
        END IF
 10   CONTINUE

      IC=Numval(bank)
      K=1

 20   IF ((IC.GE.4) .and. (K.GT.0)) THEN

        J=1
        K=0

 30     IF ((J.eq.modeln1) .or. (J.eq.modeln2) .or. (J.eq.modeln3))
     +      go to 31
        IF (J.LE.(IC-1)) THEN
          IF (VTMP(J).LT.VTMP(J+1)) THEN

            TEMP=VTMP(J)
            VTMP(J)=VTMP(J+1)
            VTMP(J+1)=TEMP
            ITMP=INDEX(J,Bank)
            INDEX(J,Bank)=INDEX(J+1,Bank)
            INDEX(J+1,Bank)=ITMP
            K=1

          END IF

 31       CONTINUE

          J=J+1
          GOTO 30

        END IF

        IC=IC-1
        GOTO 20

      END IF

      RETURN

C                          END DSORT
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
      END
```

```
      SUBROUTINE GAUSSGEN(IDUM,VLENGTH,VECTOR)
C ****************************************************
C  SUBROUTINE GAUSSGEN - uses functions GASDEV and    *
C  RAN1 to generate a vector of random variables with *
C  a normal distribution - zero mean and variance of  *
C  one.  Tests indicate that the results are good in  *
C  the one and two sigma bounds for vector lengths    *
C  of 1000 or greater.   The functions GASDEV and     *
C  RAN1 come from "Numerical Recipes", written by:    *
C  William H. Press, Brian P. Flannery, Saul A.       *
C  Teukolsky, William T. Vetterling, Cambridge Press, *
C  1986.                                              *
C                                                     *
C                                                     *
C  Subroutine GAUSSGEN                                *
C  Created: 25 July 1991                              *
C  Revised: 26 July 1991                              *
C                                                     *
C  Owner: USAF/ASD/AFIT/ENY                           *
C                                                     *
C ****************************************************
      REAL VECTOR(10),SUMVECT
      INTEGER IDUM
      INTEGER VLENGTH

C  do loop to fill array vector with random numbers in
C  a gaussian distribution

      DO I=1,VLENGTH

      VECTOR(I)=GASDEV(IDUM)
      SUMVECT = SUMVECT + VECTOR(I)
      ENDDO

C  the mean is calculated

      ZMEAN = SUMVECT/FLOAT(VLENGTH)

C  the variance is calculated

      DO I=1,VLENGTH
        ZDUM = (VECTOR(I) - ZMEAN)**2.
        SUMZDUM = SUMZDUM + ZDUM
      ENDDO
      SUMZDUM = SUMZDUM/FLOAT(VLENGTH)
      SIGMA=SQRT(SUMZDUM)


      RETURN
      END
C
      FUNCTION GASDEV(IDUM)
C ****************************************************
C  The functions GASDEV and RAN1 come from "Numerical *
C  Recipes", written by: William H. Press, Brian P.   *
C  Flannery, Saul A. Teukolsky, William T.            *
C  Vetterling, Cambridge Press, 1986.                 *
C                                                     *
C ****************************************************


      DATA ISET/0/
      IF(ISET.EQ.0)THEN
 1      V1=2.*RAN1(IDUM)-1.
        V2=2.*RAN1(IDUM)-1.
        R=V1**2 + V2**2
```

```
          IF(R.GE.1.)GO TO 1
          FAC=SQRT(-2.*LOG(R)/R)
          GSET=V1*FAC
          GASDEV=V2*FAC
          ISET=1
        ELSE
          GASDEV=GSET
          ISET=0
        ENDIF
        RETURN
        END
C
        FUNCTION RAN1(IDUM)
C ******************************************************
C  The functions GASDEV and RAN1 come from "Numerical *
C  Recipes", written by: William H. Press, Brian P.   *
C  Flannery, Saul A. Teukolsky, William T.            *
C  Vetterling, Cambridge Press, 1986.                 *
C                                                     *
C ******************************************************
        DIMENSION R(97)
        PARAMETER (M1=259200,IA1=7141,IC1=54773,RM1=3.8580247E-06)
        PARAMETER (M2=134456,IA2=8121,IC2=28411,RM2=7.4373773E-06)
        PARAMETER (M3=243000,IA3=4561,IC3=51349)
        DATA IFF/0/
        IF(IDUM.LT.0.OR.IFF.EQ.0) THEN
           IFF=1
           IX1=MOD(IC1-IDUM,M1)
           IX1=MOD(IA1*IX1+IC1,M1)
           IX2=MOD(IX1,M2)
           IX1=MOD(IA1*IX1+IC1,M1)
           IX3=MOD(IX1,M3)
           DO 11 J=1,97
              IX1 = MOD(IA1*IX1+IC1,M1)
              IX2 = MOD(IA2*IX2+IC2,M2)
              R(J)=(FLOAT(IX1)+FLOAT(IX2)*RM2)*RM1
 11        CONTINUE
           IDUM=1
         ENDIF
         IX1=MOD(IA1*IX1+IC1,M1)
         IX2=MOD(IA2*IX2+IC2,M2)
         IX3=MOD(IA3*IX3+IC3,M3)
         J=1+(97*IX3)/M3
         IF(J.GT.97.OR.J.LT.1)PAUSE
         RAN1=R(J)
         R(J)=(FLOAT(IX1)+FLOAT(IX2)*RM2)*RM1
         RETURN
         END
```

```
      SUBROUTINE GETDAT
C
      INCLUDE 'DECLARR.TXT'

C . . . . . . . . . . . . . . . . . . .
C  l o c a l   v a r i a b l e s
C . . . . . . . . . . . . . . . . . . .

      INTEGER I,J,K,L

C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

C  Read in the input data files

C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .


      CALL REDNAME
      CALL REDFLAG
      CALL REDREAL


C . . . . . . . . . . . . . . . . . . . . . . . . . . . . 

      CALL REDMAT
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . 


      Do  iq=1,NUMBANKS
       Bankname(iq)=Dfile((iq+3),iq)
      ENDDO
C      WRITE(*,*)' banknames ',BANKNAME
      RETURN

C///////////////////////////////////////////////////////////////////////
C                             END GETDAT
C======================================================================
      END
```

```
      SUBROUTINE MATML(A,B,C,L,M,N)
C This set of subroutines is being changed to sequentially reduced
C the time lag between failures.  See Subroutine GETDAT


C**********************************************************************
C THIS ROUTINE WILL MULTIPLY TWO REAL MATRICES
C A=AN L BY M MATRIX
C B=AN M BY N MATRIX
C C=THE L BY N PRODUCT OF A AND B
C NOTE ACTUAL ARGUMENT C MUST DIFFER FROM A AND B
C**********************************************************************


C/////////////////////////////////////////////////////////////////////
C//  THIS ROUTINE PROVIDES SINGLE PRECISION BOUNDS CHECKING FOR VAX
C//     OPERATION BY USING A REAL*16 VALUE FOR CALCULATING THE NEW
C//     ELEMENT IN THE C MATRIX.  IF THIS VALUE EXCEEDS THE SINGLE
C//     PRECISION LIMIT,  IT IS FORCED TO THE SINGLE PRECISION LIMIT.
      INTEGER I,J,K,L,M,N
      REAL A(L,M),B(M,N),C(L,N)
      REAL*16 CTEMP,SIGN
C_____________________________________________________________________

      DO 200 I=1,L
         DO 200 J=1,N
            CTEMP=0.
            DO 100 K=1,M
                   CTEMP=CTEMP+QEXT(A(I,K))*QEXT(B(K,J))
  100       CONTINUE
            IF(QABS(CTEMP).GT.1.Q38) THEN
               SIGN=1.
               IF(CTEMP.LT.0) SIGN=(-1.)
               CTEMP=SIGN*1.E38
            END IF
            C(I,J)=SNGLQ(CTEMP)
 200  CONTINUE
      RETURN
C/
C/////////////////////////////////////////////////////////////////////
C                                        END MATML
C---------------------------------------------------------------------
      END


      SUBROUTINE MATAD(A,B,C,L,M)
C**********************************************************************
C THIS ROUTINE ADDS TWO REAL MATRICES OF DIMENSION L BY M
C A AND B ARE THE INPUTS, C IS THE SUM
C**********************************************************************

      INTEGER I,J,L,M
      REAL A(L,M),B(L,M),C(L,M)
      DO 100 I=1,L
         DO 100 J=1,M
            C(I,J)=A(I,J)+B(I,J)
 100  CONTINUE
      RETURN
C/
C/////////////////////////////////////////////////////////////////////
C                                  END MATAD
C---------------------------------------------------------------------
      END
```

```
      SUBROUTINE MATSAV(UNIT,NAME,NR,M,N,IMG,XREAL,XIMAG,FORMT)
C Writes a file in MATRIXX readable format
C
C Parameters:
C
C   unit      INTEGER             Fortran logical unit for file
C   name      CHARACTER(*)*       Name for file.  One alpha followed by
C                                   up to 10 characters
C   nr        INTEGER             Row dimension in calling program
C   m         INTEGER             Row dimension for matrix
C   n         INTEGER             Column dimension for matrix
C   img       INTEGER             If img=0, imaginary part (ximag)
C                                   is assumed to be zero
C   xreal     DOUBLE PRECISION    Real part of the matrix to be saved
C   ximag     DOUBLE PRECISION    Imaginary part of the matrix
C   format    CHARACTER*(*)       String containing the format to be
C                                   used in writing the matrix
C                                   '(1P3E25.17)' for machine independent
C                                   '(10A8)' for fast compact
C
C
      INTEGER  UNIT,M,N,NR,IMG
      CHARACTER*(*) NAME,FORMT
      DOUBLE PRECISION XREAL(NR,*),XIMAG(NR,*)
      CHARACTER NAM*10,FORM*20
C
      NAM=NAME
      FORM=FORMT
      WRITE(UNIT,'(A10,3I5,A20)')NAM,M,N,IMG,FORM
C
C Write the real part of the matrix
C
      WRITE(UNIT,FORM) ((XREAL(I,J),I=1,M),J=1,N)
C
C Write the imaginary part of the matrix
C
      IF (IMG.NE.0) THEN
         WRITE(UNIT,FORM) ((XIMAG(I,J),I=1,M),J=1,N)
      END IF
C
      RETURN
      END
```

```
      SUBROUTINE MATSB(A,B,C,L,M)
C***********************************************************************
C THIS ROUTINE SUBTRACTS REAL MATRIX B FROM REAL MATRIX A
C DIFFERENCE IS RETURNED IN REAL MATRIX C.
C ALL THREE MATRICES ARE OF DIMENSION L BY M
C***********************************************************************
C
      INTEGER I,J,L,M
      REAL A(L,M),B(L,M),C(L,M)
      DO 100 I=1,L
         DO 100 J=1,M
            C(I,J)=A(I,J)-B(I,J)
 100  CONTINUE
      RETURN
C/
C////////////////////////////////////////////////////////////////////////
C                             END MATSB
C-----------------------------------------------------------------------
      END
```

```
      SUBROUTINE MATTP(A,B,M,N)
C***********************************************************************
C   THIS ROUTINE TRANSPOSES A MXN REAL MATRIX A AND STORES IN B
C***********************************************************************
C
      INTEGER I,J,M,N
      REAL A(M,N),B(M,N)
      DO 100 I=1,M
         DO 100 J=1,N
      PRINT*,I,J
            B(J,I)=A(I,J)
 100  CONTINUE
      RETURN
C/
C////////////////////////////////////////////////////////////////////////
C                             END MATTP
C-----------------------------------------------------------------------
      END
```

```
      SUBROUTINE MTXC21(AVEC,BVEC,IROW,ICOL,ISLICE,BANKER,IDR)

      INCLUDE 'DECLARR.TXT'

C --- ROUTINE TO CONVERT BETWEEN VECTOR AND 2 DIMENSIONAL ARRAYS
C     IF IDR=0 2D TO VECTOR CONVERSION
C     IF IDR=1 VECTOR TO 2D CONVERSION

C --- Hierarchical Modeling Variables
      Integer banker

      INTEGER IROW,ICOL,ISLICE,IDR,IR
      REAL AVEC(IROW,ICOL,BANKER),BVEC(IROW)
C______________________________________________________________________

      DO 10 IR=1,IROW
           IF(IDR.EQ.0) BVEC(IR)=AVEC(IR,ISLICE,BANKER)
           IF(IDR.EQ.1) AVEC(IR,ISLICE,BANKER)=BVEC(IR)
   10 CONTINUE
      RETURN
C//
C////////////////////////////////////////////////////////////////////////
C                             END MTXC21
C-----------------------------------------------------------------------
      END
```

```
      SUBROUTINE MTXC32(AVEC,BVEC,IROW,ICOL,IDPTH,ISLICE,BANKER,IDR)

      INCLUDE 'DECLARR.TXT'

C --- ROUTINE TO CONVERT BETWEEN 2 AND 3 DIMENSIONAL ARRAYS
C    IF IDR=0 3D TO 2D CONVERSION
C    IF IDR=1 2D TO 3D CONVERSION

C --- Hierarchical Modeling Variables
      Integer banker

      INTEGER IROW,ICOL,IDPTH,ISLICE,IDR,IC,IR
      REAL AVEC(IROW,ICOL,IDPTH,NUMBANKS),BVEC(IROW,ICOL)
C_______________________________________________________________________

      DO 10 IC=1,ICOL
          DO 10 IR=1,IROW
              IF(IDR.EQ.0) then
C               AVEC(ir,ic,islice,banker)=AVEC(ir,ic,islice,1)
                BVEC(IR,IC)=AVEC(IR,IC,ISLICE,BANKER)
                end if

              IF(IDR.EQ.1) AVEC(IR,IC,ISLICE,BANKER)=BVEC(IR,IC)


   10 CONTINUE
C        Pause 'we are returning from mtxc32'
      RETURN
C//
C///////////////////////////////////////////////////////////////////////
C                            END MTXC32
C=======================================================================
      END


      SUBROUTINE MTXDMP(A,IR,IC,NAME)
C *********************************************************************
      INTEGER IR,IC,IDX,JDX
      REAL A(IR,IC)
      CHARACTER*6 NAME
C_______________________________________________________________________

      WRITE(71,9010)NAME
      DO 10 IDX=1,IR
          IF(IC.LE.8) THEN
              WRITE(71,9000)(A(IDX,JDX),JDX=1,IC)
          ELSE
              WRITE(71,9000)(A(IDX,JDX),JDX=1,8)
              WRITE(71,9000)(A(IDX,JDX),JDX=9,IC)
          END IF
10    CONTINUE
 9000 FORMAT(' ',8(E12.5,2X))
 9010 FORMAT('0',A6,'   /////////////////////////////////////')
      RETURN
C/////////////////////////////////////////////////////////////////
C                       END MTXDMP
C=================================================================
      END
```

```
      SUBROUTINE REDFLAG
*************************************************************
*                                                           *
*     SUBROUTINE REDFLAG loads a common block of logical    *
*     flags used throughout MMAESIM to control the          *
*     program execution.  The data is loaded into an        *
*     array [ATEMP] after reading file FLAGS.DAT.  An       *
*     equivalence statement relates the values of the       *
*     [ATEMP] array with logical flag names used in the     *
*     program.                                              *
*                                                           *
*     Creation date:  8 July 1991                           *
*     Revision date:  28 July 1991                          *
*                                                           *
*     Owner: USAF/ASD/AFIT                                  *
*                                                           *
*************************************************************

      INCLUDE 'DECLARR.TXT'

C . . . . . . . . . . . . . . . . . . .
C  l o c a l   v a r i a b l e s
C . . . . . . . . . . . . . . . . . . .

C  declaration of variable types

      INTEGER ATEMP(39)

      CHARACTER*80 LINE

C  equivalence and common block statements

      EQUIVALENCE (ATEMP(1),IDID),(ATEMP(2),MODELN),
     &(ATEMP(3),MODELN1),(ATEMP(4),MODELN2),(ATEMP(5),MODELN3),
     &(ATEMP(6),ISTART),(ATEMP(7),WFLAG),(ATEMP(8),NUMFAILS),
     &(ATEMP(9),NUMBANKS),(ATEMP(10),BANK),(ATEMP(11),HIERARCHY),
     &(ATEMP(12),XITER),(ATEMP(13),SENSORBIAS),(ATEMP(14),FLTRT0),
     &(ATEMP(15),NFLTR(1)),(ATEMP(16),NFLTR(2)),(ATEMP(17),NFLTR(3)),
     &(ATEMP(18),NFLTR(4)),(ATEMP(19),NFLTR(5)),(ATEMP(20),NFLTR(6)),
     &(ATEMP(21),NFLTR(7)),(ATEMP(22),NFLTR(8)),(ATEMP(23),NFLTR(9)),
     &(ATEMP(24),NFLTR(10)),(ATEMP(25),NFLTR(11)),(ATEMP(26),NFLTR(12)),
     &(ATEMP(27),NFLTR(13)),(ATEMP(28),NFLTR(14)),(ATEMP(29),NFLTR(15)),
     &(ATEMP(30),BANKFLAG),(ATEMP(31),INITV2),(ATEMP(32),INITV),
     &(ATEMP(33),IACTORDR),(ATEMP(34),BETAFLG),(ATEMP(35),ITRIMZ),
     &(ATEMP(36),ISLCT),(ATEMP(37),DSEED),(ATEMP(38),IACTFAIL),
     &(ATEMP(39),IACTFL2)

C  initialization of variables

      ICOUNTR = 0

C  main program

      OPEN(UNIT=11,FILE='FLAGS.DAT',STATUS='UNKNOWN')
      DO WHILE(ICOUNTR.LT.39)
        READ(11,'(A)') LINE
        IF(LINE(1:1).EQ.' ')THEN
          ICOUNTR = ICOUNTR+1
          LOCATION = INDEX(LINE,'=')+1
          READ(LINE(LOCATION:),*)ATEMP(ICOUNTR)
        ENDIF
      ENDDO

C  output check - used for debugging

C     WRITE(*,*) ATEMP
```

```
c        WRITE(*,*)IDID,MODELN,MODELN1,MODELN2,MODELN3,ISTART,WFLAG,
c       &NUMFAILS,NUMBANKS,BANK,HIERARCHY,XITER,WGNFAC,PRBMIN,
c       &SENSORBIAS,FLTRT0,NFILT(1),NFILT(2),NFILT(3),NFILT(4),NFILT(5),
c       &NFILT(6),NFILT(7),NFILT(8),NFILT(9),NFILT(10),NFILT(11),
c       &NFILT(12),NFILT(13),NFILT(14),NFILT(15),BANKFLAG,INITV2,INITV,
c       &IPAR,LIW,LRW,INFO(1),INFO(2),INFO(3),INFO(4),IWORK,RPAR,RWORK,
c       &IACTORDR,M,BETAFLG,ITRIMZ,ISLCT,IACTFAIL,IACTFL2
C        Write(*,*) iactordr,m,betaflg,itrimz,islct,iactfail

        CLOSE(11)
        RETURN
        END
```

```
      SUBROUTINE REDMAT
*********************************************************
*                                                       *
*   SUBROUTINE REDMAT is called from subroutine GETDAT  *
*   and loads into a common block the data arrays for   *
*   the [F] matrix - (ZA), the [B] matrix - (ZB), the   *
*   state transition matrix [PHI] - ZPHI, ...           *
*                                                       *
*                                                       *
*   CREATION DATE:  12 JULY 1991                        *
*   REVISION DATE:  22 JULY 1991                        *
*                                                       *
*   OWNER: USAF/ASD/AFIT -  Wright Patterson AFB        *
*                                                       *
*********************************************************

      INCLUDE 'DECLARR.TXT'

C . . . . . . . . . . . . . . . . . . .
C  l o c a l   v a r i a b l e s
C . . . . . . . . . . . . . . . . . . .

      CHARACTER*80 LINE

      LINE = 'C'

C . . . . . . . . . . . . . . . . . . .


      DO IBANKZ=1,NUMBANKS
        L=IBANKZ

        DO K=1,NFLTR(IBANKZ)

           OPEN(UNIT=9,FILE=DFILE(K,IBANKZ),STATUS='OLD')
C           WRITE(*,*)' BANK = ',IBANKZ,' FILTER = ',K
C           WRITE(*,*)'DFILE = ',DFILE(K,IBANKZ)

          DO I=1,7
            READ(9,'(A132)')LINE
          ENDDO

          READ(9,*)((ZA(I,J,K,L),I=1,8),J=1,8)
C
          READ(9,'(A132)')LINE
C
          READ(9,*)((ZB(I,J,K,L),I=1,8),J=1,6)
C
          READ(9,'(A132)')LINE
C
          READ(9,*)((ZPHIX(I,J,K,L),I=1,14),J=1,14)
C
          READ(9,'(A132)')LINE
C
          READ(9,*)((ZBD(I,J,K,L),I=1,14),J=1,6)
C
          READ(9,'(A132)')LINE
C
          READ(9,*)((ZCQDCN(I,J,K,L),I=1,8),J=1,8)
C
          READ(9,'(A132)')LINE
C
          READ(9,*)((ZH(I,J,K,L),I=1,7),J=1,14)
C          WRITE(*,*)' H = ',ZH(4,1,K,L)
C
```

```
          READ(9,'(A132)')LINE
C
          READ(9,*)((ZGKF(I,J,K,L),I=1,14),J=1,7)
C
          READ(9,'(A132)')LINE
C
          READ(9,*)((ZR(I,J,K,L),I=1,7),J=1,7)
C          WRITE(*,*)' R = ',ZR(6,6,K,L)
C
          READ(9,'(A132)')LINE
C
          READ(9,*)((ZAK(I,J,K,L),I=1,7),J=1,7)
C          WRITE(*,*)' ZAK = ',ZAK(7,7,K,L)
C
          READ(9,'(A132)')LINE
C
          READ(9,*)((ZAKINV(I,J,K,L),I=1,7),J=1,7)
C          WRITE(*,*)' ZAKINV = ',ZAKINV(7,7,K,L)
C
          READ(9,'(A132)')LINE
C
          READ(9,*)ZDETAK(K,L)
C          WRITE(*,*)ZDETAK(K,L)
C

            ENDDO
      ENDDO

      CLOSE(9)
      RETURN
      END
```

```
      SUBROUTINE REDREAL
*************************************************************
*                                                           *
*     SUBROUTINE REDREAL loads a common block of real       *
*     flags used throughout MMAESIM to control the          *
*     program execution.  The data is loaded into an        *
*     array [ATEMP] after reading file FLAGS.DAT.  An       *
*     equivalence statement relates the values of the       *
*     [ATEMP] array with logical flag names used in the     *
*     program.                                              *
*                                                           *
*     Creation date:  8 July 1991                           *
*     Revision date:  9 July 1991                           *
*                                                           *
*     Owner: USAF/ASD/AFIT                                  *
*                                                           *
*************************************************************

      INCLUDE 'DECLARR.TXT'

C . . . . . . . . . . . . . . . . . . .
C  l o c a l   v a r i a b l e s
C . . . . . . . . . . . . . . . . . . .

      CHARACTER*80 LINE
      REAL RTEMP(22)
      DOUBLE PRECISION DPTEMP(1)

      EQUIVALENCE (RTEMP(1),TSAMP),(RTEMP(2),PRBMIN),
     &(RTEMP(3),PRBFLTRTO),(RTEMP(4),DSIM),(RTEMP(5),WGNFAC),
     &(RTEMP(6),ZBIASAMNT(1)),(RTEMP(7),ZBIASAMNT(2)),
     &(RTEMP(8),ZBIASAMNT(3)),(RTEMP(9),ZBIASAMNT(4)),
     &(RTEMP(10),ZBIASAMNT(5)),(RTEMP(11),ZBIASAMNT(6)),
     &(RTEMP(12),ZBIASAMNT(7)),(RTEMP(13),TIMELAG1),
     &(RTEMP(14),TIMELAG2),(RTEMP(15),TSIG11),(RTEMP(16),
     &TSIG22),(RTEMP(17),TSIG33),(RTEMP(18),TSIG44),(RTEMP(19),
     &TSIG55),(RTEMP(20),TSIG66),(RTEMP(21),TSIG77),(RTEMP(22),
     &TSIG88)

      EQUIVALENCE (DPTEMP(1),M)


      ICOUNT = 0
      OPEN(UNIT=12,FILE='REALS.DAT',STATUS='UNKNOWN')
      DO WHILE(ICOUNT.LT.22)
        READ(12,'(A)') LINE
        IF(LINE(1:1).EQ.' ')THEN
          ICOUNT = ICOUNT+1
          LOCATION = INDEX(LINE,'=')+1
          READ(LINE(LOCATION:),*)RTEMP(ICOUNT)
        ENDIF
      ENDDO
      ICOUNT = 0
      DO WHILE(ICOUNT.LT.1)
        READ(12,'(A)') LINE
        IF(LINE(1:1).EQ.' ')THEN
          ICOUNT = ICOUNT+1
          LOCATION = INDEX(LINE,'=')+1
          READ(LINE(LOCATION:),*)DPTEMP(ICOUNT)
        ENDIF
      ENDDO

C     output section used for debugging
C      WRITE(*,*)'TSIG11,TSIG88 ',TSIG11,TSIG88
C      WRITE(*,*)'M = ',M
      CLOSE(12)
      RETURN
      END
```

```
      SUBROUTINE REDNAME
***************************************************************
*                                                             *
*     SUBROUTINE REDNAME loads a common block of character    *
*     names used throughout MMAESIM to control the            *
*     reading of the proper data files.  The filter names     *
*     are stored in the DFILE array and loaded into a         *
*     common block.                                           *
*                                                             *
*                                                             *
*     Creation date:  10 July 1991                            *
*     Revision date:  11 July 1991                            *
*                                                             *
*     Owner: USAF/ASD/AFIT                                    *
*                                                             *
***************************************************************

      INCLUDE 'DECLARR.TXT'

C . . . . . . . . . . . . . . . . . . . . . .
C  l o c a l   v a r i a b l e s
C . . . . . . . . . . . . . . . . . . . . . .

C  declaration of variable types

      INTEGER ICOUNTR,JCOUNTR
      CHARACTER*80 LINE

C  equivalence and common block statements
C

C  initialization of variables

      ICOUNTR = 0
      JCOUNTR = 0
C  main program

      OPEN(UNIT=13,FILE='FLTRNAME.DAT',STATUS='UNKNOWN')
      DO WHILE(JCOUNTR.LT.15)
        JCOUNTR = JCOUNTR + 1
        DO WHILE(ICOUNTR.LT.20)
          READ(13,'(A)') LINE
          IF(LINE(1:1).EQ.' ')THEN
            ICOUNTR = ICOUNTR+1
            LOCATION = INDEX(LINE,'=')+2
            READ(LINE(LOCATION:),'(A)')DFILE(ICOUNTR,JCOUNTR)
C            WRITE(*,*)DFILE(ICOUNTR,JCOUNTR)
          ENDIF
        ENDDO
        ICOUNTR = 0
      ENDDO

C  output check - used for debugging

      CLOSE(13)
      RETURN
      END
```

```
      SUBROUTINE SMUL(A,B,C,L,M)
C*********************************************************************
C THIS ROUTINE MULTIPLIES A REAL MATRIX BY A REAL SCALAR
C A= THE SCALAR
C B= THE MATRIX
C C= THE PRODUCT
C B AND C ARE OF DIMENSION L BY M
C*********************************************************************
C
      INTEGER I,J,L,M
      REAL A,B(L,M),C(L,M)
      DO 100 I=1,L
         DO 100 J=1,M
            C(I,J)=A*B(I,J)
 100  CONTINUE
      RETURN
C/
C///////////////////////////////////////////////////////////////////////
C                            END SMUL
C-----------------------------------------------------------------------
      END
```

```
      SUBROUTINE TEMPORAL(RSSAVE,TEMPAVG)
*//////////////////////////////////////////////////////////*
*  Calculates the temporal average of the residuals       *
*  within each of the elemental filters                   *
      *//////////////////////////////////////////////////////////*

      Real Tmpavg,Tmpavga,Tmpavgb,Tmpavgc,Tmpavgd
      Real Tempavg(192,7,13),Temphold
      Integer IIMOD,IZ,JZ

      Include 'Declarr.txt'


      Do IIMOD = 1,13

        Do JZ = 1,7

          Do IZ = 1,64
            Tmpavg = RSSAVE(IZ,JZ,IIMOD) + Tmpavg
          Enddo
          Tmpavga = Tmpavg
          Do IZ = 65,128
            Tmpavg = RSSAVE(IZ,JZ,IIMOD) + Tmpavg
          Enddo
          Tmpavgb = Tmpavg
          Do IZ = 129,192
            Tmpavg = RSSAVE(IZ,JZ,IIMOD) + Tmpavg
          Enddo
          Tmpavgc = Tmpavg

          Do IZ = 1,192

            Temphold = (Float(IZ)/64.0) + 1.0

            If(IZ.le.64)then
              Tempavg(TEMPHOLD,JZ,IIMOD) = Tmpavga
            Endif

            If((IZ.gt.64).and.(IZ.le.128))then
              Tempavg(TEMPHOLD,JZ,IIMOD) = Tmpavgb
            Endif

            If((IZ.gt.128).and.(IZ.le.192))then
              Tempavg(TEMPHOLD,JZ,IIMOD) = Tmpavgc
            Endif


          Enddo
        Enddo
      Enddo

      Return
*//////////////////////////////////////////////////////////*
*                                                          *
*//////////////////////////////////////////////////////////*
      End
```

APPENDIX D: MATRIX$_X$ MACROS

These macros were used to create the filter files and plot the results. Matrix$_x$ [15] resides on the Flight Dynamics VAX computer. The routines are included in the end of this appendix. Each of the routines are described within this appendix.

The Filecreate routine accessess the other Matrix$_x$ routines. This routine calls MTX.MXX and the SETUPXX routines. The SETUPXX routines set up each of the appropriate banks by operating on the data loaded within Matrix$_x$. The SETUPXX routines include the hypothesized failure for each bank and filter. Each filter within bank 1 hypothesizes a single failure. Bank 2 assumes a left stabilator failure. Thus every filter within bank 2 assumes a left stabilator failure and another failure corresponding to the filter designation. The bank numbers range from B1 - B9, and then the designations change to X0 - X3. Data files for the fully-functional aircraft model elemental filter for the MMAE can be found in Appendix E. A MMAE simulation users manual guides the reader through the design and running of the simulation. The users manual can be obtained through Dr. Peter Maybeck, Department of Electrical Engineering, Air Force Institute of Technology, WPAFB, OH.

```
//
//        FILECREATE.MXX MATRIXX EXECUTABLE MACRO
//
//                Author:  Capt Gregory L. Stratton
//          Date created:  25 August 1991
//          Date revised:  27 September 1991
//
//  This macro creates every filter file for each bank for the
//  MMAESIM program.  Below are listed the variables that need to
//  reside in MATRIXX memory before execution:
//
//     forig,    The 8x8 plant matrix (A matrix)
//     borig,    The 8x6 control matrix (B matrix)
//     horig,    The 7x14 measurement equation matrix (H matrix)
//     r,        The 7x7 sensor covariance matrix
//     q,        The 6x6 white Gaussian noise covariance matrix
//     g,        The 8x6 white noise multiplier matrix
//
//  This macro calls the macros "setupbn.mxx" (where n is the bank
//  number), which in turn creates the files for that particular bank.
//  In all, 13 filters for each of the 13 banks are created.
//
//  Each filter file is set up so that it may also be copied into files
//  F01Bn.dat, F02Bn.dat, or F03Bn.dat and used as truth models of hard
//  failures.  Soft failure truth models must be generated separately.
//
//
//  S T A R T  M A C R O
//
//
//  Set up initial constants
//
n=14;
delt=1/64;
rz14=0*ones(1,14);
cz8=0*ones(8,1);
cz14=0*ones(14,1);
act=diag([20.2,20.2,20.2,20.2,20.2,16]);
aa=forig;
baug=[0*ones(8,6);act];
faug=[aa,borig;0*ones(6,8),-act];
g=[g;0*ones(6,6)];
gd=eye(14);
//
//  Set up F, B, and H matrices for bank 1 filters
//
bank=1;
fcon=faug;
bcon=baug;
bbcon=borig;
hcon=horig;
exec('setupb1.mxx')
//
//return
//
//  Set up F, B, and H matrices for bank 2 filters
//  where actuator 1 has been determined to be failed
//
bank=2;
fcon=faug;
//fcon(:,9)=cz14;
bcon=baug;
b(:,1)=cz14;
bbcon=borig;
bbcon(:,1)=cz8;
hcon=horig;
```

```
exec('setupb2.mxx')
//
//  Set up F, B, and H matrices for bank 3 filters
//  where actuator 2 has been determined to be failed
//
bank=3;
fcon=faug;
//fcon(:,10)=cz14;
bcon=baug;
b(:,2)=cz14;
bbcon=borig;
bbcon(:,2)=cz8;
hcon=horig;
exec('setupb3.mxx')
//
//  Set up F, B, and H matrices for bank 4 filters
//  where actuator 3 has been determined to be failed
//
bank=4;
fcon=faug;
//fcon(:,11)=cz14;
bcon=baug;
b(:,3)=cz14;
bbcon=borig;
bbcon(:,3)=cz8;
hcon=horig;
exec('setupb4.mxx')
//
//  Set up F, B, and H matrices for bank 5 filters
//  where actuator 4 has been determined to be failed
//
bank=5;
fcon=faug;
//fcon(:,12)=cz14;
bcon=baug;
b(:,4)=cz14;
bbcon=borig;
bbcon(:,4)=cz8;
hcon=horig;
exec('setupb5.mxx')
//
//  Set up F, B, and H matrices for bank 6 filters
//  where actuator 5 has been determined to be failed
//
bank=6;
fcon=faug;
//fcon(:,13)=cz14;
bcon=baug;
b(:,5)=cz14;
bbcon=borig;
bbcon(:,5)=cz8;
hcon=horig;
exec('setupb6.mxx')
//
//  Set up F, B, and H matrices for bank 7 filters
//  where sensor 1 has been determined to be failed
//
bank=7;
fcon=faug;
bcon=baug;
bbcon=borig;
hcon=horig;
hcon(1,:)=rz14;
exec('setupb7.mxx')
//
//  Set up F, B, and H matrices for bank 8 filters
```

```
//  where sensor 2 has been determined to be failed
//
bank=8;
fcon=faug;
bcon=baug;
bbcon=borig;
hcon=horig;
hcon(2,:)=rz14;
exec('setupb8.mxx')
//
//  Set up F, B, and H matrices for bank 9 filters
//  where sensor 3 has been determined to be failed
//
bank=9;
fcon=faug;
bcon=baug;
bbcon=borig;
hcon=horig;
hcon(3,:)=rz14;
exec('setupb9.mxx')
//
//  Set up F, B, and H matrices for bank 10 filters
//  where sensor 4 has been determined to be failed
//
bank=10;
fcon=faug;
bcon=baug;
bbcon=borig;
hcon=horig;
hcon(4,:)=rz14;
exec('setupx0.mxx')
//
//  Set up F, B, and H matrices for bank 11 filters
//  where sensor 5 has been determined to be failed
//
bank=11;
fcon=faug;
bcon=baug;
bbcon=borig;
hcon=horig;
hcon(5,:)=rz14;
exec('setupx1.mxx')
//
//  Set up F, B, and H matrices for bank 12 filters
//  where sensor 6 has been determined to be failed
//
bank=12;
fcon=faug;
bcon=baug;
bbcon=borig;
hcon=horig;
hcon(6,:)=rz14;
exec('setupx2.mxx')
//
//  Set up F, B, and H matrices for bank 13 filters
//  where sensor 7 has been determined to be failed
//
bank=13;
fcon=faug;
bcon=baug;
bbcon=borig;
hcon=horig;
hcon(7,:)=rz14;
exec('setupx3.mxx')
//
//  Set up F, B, and H matrices for bank 7 filters
```

```
//  where actuator 6 has been determined to be failed
//
//  This section has been commented out
//  Originally this generated bank 7 filters
//  It is now set to be bank 14 if ever used
//
//bank=14;
//fcon=faug;
////fcon(:,14)=cz14;
//bcon=baug;
//b(:,6)=cz14;
//bbcon=borig;
//bbcon(:,6)=cz8;
//hcon=horig;
//exec('setupx4.mxx')
//
//
return
//
```

```
// RESIDUAL PLOTTING MACRO for MMAESIM RESULTS
//
// This macro plots residual results form mmaesim quickly
// and generates files for printing at a future date.
//
rsff=RS(:,1:7);
bdup=BDU(:,1:7);
bdlw=-1.0*bdup;
//rsls=RS(:,8:14);
//bduls=BDU(:,8:14);
//bdlls=-1.0*BDU(:,8:14);
//rsrs=RS(:,15:21);
//bdurs=BDU(:,15:21);
//bdlrs=-1.0*BDU(:,15:21);
//rslf=RS(:,22:28);
//bdulf=BDU(:,22:28);
//bdllf=-1.0*BDU(:,22:28);
//rsrf=RS(:,29:35);
//bdurf=BDU(:,29:35);
//bdlrf=-1.0*BDU(:,29:35);
//rsrd=RS(:,36:42);
//bdurd=BDU(:,36:42);
//bdlrd=-1.0*BDU(:,36:42);
//
// fully functional filter
//
rsv1=[bdup(:,1),bdlw(:,1),rsff(:,1)]
rsv2=[bdup(:,2),bdlw(:,2),rsff(:,2)]
rsv3=[bdup(:,3),bdlw(:,3),rsff(:,3)]
rsv4=[bdup(:,4),bdlw(:,4),rsff(:,4)]
rsv5=[bdup(:,5),bdlw(:,5),rsff(:,5)]
rsv6=[bdup(:,6),bdlw(:,6),rsff(:,6)]
rsv7=[bdup(:,7),bdlw(:,7),rsff(:,7)]
//
//
//
plot(ts,rsv1,'title/Velocity Sensor cs01/...
xlabel/time (seconds)/...
ylabel/residual value/');
hard('residplt1.dat')
//
plot(ts,rsv2,'title/Angle of Attack Sensor cs01/...
xlabel/Time (seconds)/...
ylabel/residual value/');
hard('residplt2.dat')
//
plot(ts,rsv3,'title/Pitch Rate Sensor cs01/...
xlabel/Time (seconds)/...
ylabel/residual value/');
hard('residplt3.dat')
//
plot(ts,rsv4,'title/Normal Acceleration Sensor cs01/...
xlabel/Time (seconds)/...
ylabel/residual value/');
hard('residplt4.dat')
//
plot(ts,rsv5,'title/Roll Rate Sensor cs01/...
xlabel/Time (seconds)/...
ylabel/residual value/');
hard('residplt5.dat')
//
plot(ts,rsv6,'title/Yaw Rate Sensor cs01/...
xlabel/Time (seconds)/...
ylabel/residual value/');
hard('residplt6.dat')
//
```

```
plot(ts,rsv7,'title/Lateral Acceleration Sensor cs01/...
xlabel/Time (seconds)/...
ylabel/residual value/');
hard('residplt7.dat')
//
return
//
// left stabilator
//
rsv1=[rsls(:,1),bduls(:,1),bdlls(:,1)]
rsv2=[rsls(:,2),bduls(:,2),bdlls(:,2)]
rsv3=[rsls(:,3),bduls(:,3),bdlls(:,3)]
rsv4=[rsls(:,4),bduls(:,4),bdlls(:,4)]
rsv5=[rsls(:,5),bduls(:,5),bdlls(:,5)]
rsv6=[rsls(:,6),bduls(:,6),bdlls(:,6)]
rsv7=[rsls(:,7),bduls(:,7),bdlls(:,7)]
//
plot(ts,rsv1,'title/Velocity Sensor cs02/...
xlabel/time (seconds)/...
ylabel/residual value/');
hard('residplt8.dat')
//
plot(ts,rsv2,'title/Angle of Attack Sensor cs02/...
xlabel/Time (seconds)/...
ylabel/residual value/');
hard('residplt9.dat')
//
plot(ts,rsv3,'title/Pitch Rate Sensor cs02/...
xlabel/Time (seconds)/...
ylabel/residual value/');
hard('residplt10.dat')
//
plot(ts,rsv4,'title/Normal Acceleration Sensor cs02/...
xlabel/Time (seconds)/...
ylabel/residual value/');
hard('residplt11.dat')
//
plot(ts,rsv5,'title/Roll Rate Sensor cs02/...
xlabel/Time (seconds)/...
ylabel/residual value/');
hard('residplt12.dat')
//
plot(ts,rsv6,'title/Yaw Rate Sensor cs02/...
xlabel/Time (seconds)/...
ylabel/residual value/');
hard('residplt13.dat')
//
plot(ts,rsv7,'title/Lateral Acceleration Sensor cs02/...
xlabel/Time (seconds)/...
ylabel/residual value/');
hard('residplt14.dat')
//
// right stabilator
//
rsv1=[rsrs(:,1),bdurs(:,1),bdlrs(:,1)]
rsv2=[rsrs(:,2),bdurs(:,2),bdlrs(:,2)]
rsv3=[rsrs(:,3),bdurs(:,3),bdlrs(:,3)]
rsv4=[rsrs(:,4),bdurs(:,4),bdlrs(:,4)]
rsv5=[rsrs(:,5),bdurs(:,5),bdlrs(:,5)]
rsv6=[rsrs(:,6),bdurs(:,6),bdlrs(:,6)]
rsv7=[rsrs(:,7),bdurs(:,7),bdlrs(:,7)]
//
plot(ts,rsv1,'title/Velocity Sensor cs03/...
xlabel/time (seconds)/...
ylabel/residual value/');
hard('residplt15.dat')
```

```
//
plot(ts,rsv2,'title/Angle of Attack Sensor cs03/...
xlabel/Time (seconds)/...
ylabel/residual value/');
hard('residplt16.dat')
//
plot(ts,rsv3,'title/Pitch Rate Sensor cs03/...
xlabel/Time (seconds)/...
ylabel/residual value/');
hard('residplt17.dat')
//
plot(ts,rsv4,'title/Normal Acceleration Sensor cs03/...
xlabel/Time (seconds)/...
ylabel/residual value/');
hard('residplt18.dat')
//
plot(ts,rsv5,'title/Roll Rate Sensor cs03/...
xlabel/Time (seconds)/...
ylabel/residual value/');
hard('residplt19.dat')
//
plot(ts,rsv6,'title/Yaw Rate Sensor cs03/...
xlabel/Time (seconds)/...
ylabel/residual value/');
hard('residplt20.dat')
//
plot(ts,rsv7,'title/Lateral Acceleration Sensor cs03/...
xlabel/Time (seconds)/...
ylabel/residual value/');
hard('residplt21.dat')
//
//  left flaperon
//
rsv1=[rslf(:,1),bdulf(:,1),bdllf(:,1)]
rsv2=[rslf(:,2),bdulf(:,2),bdllf(:,2)]
rsv3=[rslf(:,3),bdulf(:,3),bdllf(:,3)]
rsv4=[rslf(:,4),bdulf(:,4),bdllf(:,4)]
rsv5=[rslf(:,5),bdulf(:,5),bdllf(:,5)]
rsv6=[rslf(:,6),bdulf(:,6),bdllf(:,6)]
rsv7=[rslf(:,7),bdulf(:,7),bdllf(:,7)]
//
plot(ts,rsv1,'title/Velocity Sensor cs04/...
xlabel/time (seconds)/...
ylabel/residual value/');
hard('residplt22.dat')
//
plot(ts,rsv2,'title/Angle of Attack Sensor cs04/...
xlabel/Time (seconds)/...
ylabel/residual value/');
hard('residplt23.dat')
//
plot(ts,rsv3,'title/Pitch Rate Sensor cs04/...
xlabel/Time (seconds)/...
ylabel/residual value/');
hard('residplt24.dat')
//
plot(ts,rsv4,'title/Normal Acceleration Sensor cs04/...
xlabel/Time (seconds)/...
ylabel/residual value/');
hard('residplt25.dat')
//
plot(ts,rsv5,'title/Roll Rate Sensor cs04/...
xlabel/Time (seconds)/...
ylabel/residual value/');
hard('residplt26.dat')
//
```

```
plot(ts,rsv6,'title/Yaw Rate Sensor cs04/...
xlabel/Time (seconds)/...
ylabel/residual value/');
hard('residplt27.dat')
//
plot(ts,rsv7,'title/Lateral Acceleration Sensor cs04/...
xlabel/Time (seconds)/...
ylabel/residual value/');
hard('residplt28.dat')
//
//  right flaperon
//
rsv1=[rsrf(:,1),bdurf(:,1),bdlrf(:,1)]
rsv2=[rsrf(:,2),bdurf(:,2),bdlrf(:,2)]
rsv3=[rsrf(:,3),bdurf(:,3),bdlrf(:,3)]
rsv4=[rsrf(:,4),bdurf(:,4),bdlrf(:,4)]
rsv5=[rsrf(:,5),bdurf(:,5),bdlrf(:,5)]
rsv6=[rsrf(:,6),bdurf(:,6),bdlrf(:,6)]
rsv7=[rsrf(:,7),bdurf(:,7),bdlrf(:,7)]
//
plot(ts,rsv1,'title/Velocity Sensor cs05/...
xlabel/time (seconds)/...
ylabel/residual value/');
hard('residplt29.dat')
//
plot(ts,rsv2,'title/Angle of Attack Sensor cs05/...
xlabel/Time (seconds)/...
ylabel/residual value/');
hard('residplt30.dat')
//
plot(ts,rsv3,'title/Pitch Rate Sensor cs05/...
xlabel/Time (seconds)/...
ylabel/residual value/');
hard('residplt31.dat')
//
plot(ts,rsv4,'title/Normal Acceleration Sensor cs05/...
xlabel/Time (seconds)/...
ylabel/residual value/');
hard('residplt32.dat')
//
plot(ts,rsv5,'title/Roll Rate Sensor cs05/...
xlabel/Time (seconds)/...
ylabel/residual value/');
hard('residplt33.dat')
//
plot(ts,rsv6,'title/Yaw Rate Sensor cs05/...
xlabel/Time (seconds)/...
ylabel/residual value/');
hard('residplt34.dat')
//
plot(ts,rsv7,'title/Lateral Acceleration Sensor cs05/...
xlabel/Time (seconds)/...
ylabel/residual value/');
hard('residplt35.dat')
//
//  rudder
//
rsv1=[rsrd(:,1),bdurd(:,1),bdlrd(:,1)]
rsv2=[rsrd(:,2),bdurd(:,2),bdlrd(:,2)]
rsv3=[rsrd(:,3),bdurd(:,3),bdlrd(:,3)]
rsv4=[rsrd(:,4),bdurd(:,4),bdlrd(:,4)]
rsv5=[rsrd(:,5),bdurd(:,5),bdlrd(:,5)]
rsv6=[rsrd(:,6),bdurd(:,6),bdlrd(:,6)]
rsv7=[rsrd(:,7),bdurd(:,7),bdlrd(:,7)]
//
plot(ts,rsv1,'title/Velocity Sensor cs06/...
```

```
xlabel/time (seconds)/...
ylabel/residual value/');
hard('residplt36.dat')
//
plot(ts,rsv2,'title/Angle of Attack Sensor cs06/...
xlabel/Time (seconds)/...
ylabel/residual value/');
hard('residplt37.dat')
//
plot(ts,rsv3,'title/Pitch Rate Sensor cs06/...
xlabel/Time (seconds)/...
ylabel/residual value/');
hard('residplt38.dat')
//
plot(ts,rsv4,'title/Normal Acceleration Sensor cs06/...
xlabel/Time (seconds)/...
ylabel/residual value/');
hard('residplt39.dat')
//
plot(ts,rsv5,'title/Roll Rate Sensor cs06/...
xlabel/Time (seconds)/...
ylabel/residual value/');
hard('residplt40.dat')
//
plot(ts,rsv6,'title/Yaw Rate Sensor cs06/...
xlabel/Time (seconds)/...
ylabel/residual value/');
hard('residplt41.dat')
//
plot(ts,rsv7,'title/Lateral Acceleration Sensor cs06/...
xlabel/Time (seconds)/...
ylabel/residual value/');
hard('residplt42.dat')
//
```

```
//  This Macro plots results from MMAESIM that are thesis
//  quality to be used for printing on the LN03 laser printer.
//
pr1=prb(:,4:9);
pr2=prb(:,10:16);
//
plot(ti,pr1,'strip ...
title/Probability, Left_Stabilator_Failure/ ...
ymin/0/ ymax/1/ ...
xlabel/Time (seconds)/ ...
ylabel/FF|A1|A2|A3|A4|A5/')
hard('pl1a1_hfsl.dat')
//
plot(ti,pr2,'strip ...
title/Probability, Left_Stabilator_Failure/ ...
ymin/0/ ymax/1/ ...
xlabel/Time (seconds)/ ...
ylabel/S1|S2|S3|S4|S5|S6|S7/')
hard('pl2a1_hfsl.dat')
//
//sg1=[st(:,1:8),gs(:,1:2)];
sg1=[st(:,1:4),gs(:,1)];
sg2=[st(:,5:8),gs(:,2)];
//
rtod=57.29578;
sg1(:,1)=rtod*sg1(:,1);
sg1(:,3:4)=rtod*sg1(:,3:4);
sg2(:,1:4)=rtod*sg2(:,1:4);
//
plot(ti,sg1,'strip ...
title/States, Left_Stabilator_Failure/ ...
xlabel/Time (seconds)/ ...
ylabel/theta|u|alpha|q|Ancg/')
hard('pl3a1_hfsl.dat')
//
plot(ti,sg2,'strip ...
title/States, Left_Stabilator_Failure/ ...
xlabel/Time (seconds)/ ...
ylabel/phi|beta|p|r|Aycg/')
hard('pl4a1_hfsl.dat')
//
```

```
//  This Macro plots results from MMAESIM that are thesis
//  quality to be used for printing on the LN03 laser printer.
//
pr1=prb(:,4:9);
pr2=prb(:,10:16);
//
plot(ti,pr1,'strip ...
title/Probability, No_Failure scenario, / ...
ymin/0/ ymax/1/ ...
xlabel/Time (seconds)/ ...
ylabel/FF|A1|A2|A3|A4|A5/')
hard('pl1ff_hfsl.dat')
//
plot(ti,pr2,'strip ...
title/Probability, No_Failure scenario/ ...
ymin/0/ ymax/1/ ...
xlabel/Time (seconds)/ ...
ylabel/S1|S2|S3|S4|S5|S6|S7/')
hard('pl2ff_hfsl.dat')
//
//sg1=[st(:,1:8),gs(:,1:2)];
sg1=[st(:,1:4),gs(:,1)];
sg2=[st(:,5:8),gs(:,2)];
//
rtod=57.29578;
sg1(:,1)=rtod*sg1(:,1);
sg1(:,3:4)=rtod*sg1(:,3:4);
sg2(:,1:4)=rtod*sg2(:,1:4);
//
plot(ti,sg1,'strip ...
title/States, No_Failure scenario/ ...
xlabel/Time (seconds)/ ...
ylabel/theta|u|alpha|q|Ancg/')
hard('pl3ff_hfsl.dat')
//
plot(ti,sg2,'strip ...
title/States, No_Failure scenario/ ...
xlabel/Time (seconds)/ ...
ylabel/phi|beta|p|r|Aycg/')
hard('pl4ff_hfsl.dat')
//
```

```
//
GQGT = G * Q * G';
FF2 = F;
FF2(n+1,n+1) = 0.;
PHI = split(disc(FF2,n,delt),n);
PGQGPT = PHI * GQGT * PHI';
Qd = (PGQGPT + GQGT) * 0.5 * delt;
clear GQGT PGQGPT FF2;
//
qtemp = Gd*Qd*Gd';
temp77 = [PHI'+H'/R*H/PHI*qtemp, -H'/R*H/PHI; -PHI\qtemp,inv(PHI')'];
[vtemp,dtemp] = eig(temp77);
dtemp = diag(dtemp);
idx77 = sort(abs(dtemp));
chi77 = vtemp(1:n,idx77(1:n));
lamda7 = vtemp((n+1):(2*n),idx77(1:n));
stemp = lamda7/chi77;
Pmss = stemp';
Kss = stemp*H'/(H*stemp*H' + R);
Ppss = PHI\(Pmss-Gd*Qd*Gd')/PHI';
Pmss = real(Pmss);
Kss = real(Kss);
Ppss = real(Ppss);
clear dtemp idx77 chi77 lamda7 stemp temp77 vtemp qtemp;
gkf=kss;
phix=phi;
bd=.5*(phi*b+b)*delt;
cqd=(chol(qd(1:8,1:8)))';
ak=h*pmss*h'+r;
akinv=inv(ak);
detak=det(ak);
return;
//
//
```

```
display('                         QWICK        KFEVAL')
display(' ')
display('              A Kalman Filter Performance Evaluation Tool')
display('                                  by')
display('                          Peter S. Maybeck')
display(' ')
display('                Version VAX1.2    ALL RIGHTS RESERVED')
display(' ')
//
GQGT = G * Q * G';
FF2 = F;
FF2(n+1,n+1) = 0.;
PHI = split(disc(FF2,n,delt),n);
PGQGPT = PHI * GQGT * PHI';
Qd = (PGQGPT + GQGT) * 0.5 * delt;
clear GQGT PGQGPT FF2;
//
qtemp = Gd*Qd*Gd';
temp77 = [PHI'+H'/R*H/PHI*qtemp, -H'/R*H/PHI; -PHI\qtemp,inv(PHI')'];
[vtemp,dtemp] = eig(temp77);
dtemp = diag(dtemp);
idx77 = sort(abs(dtemp));
chi77 = vtemp(1:n,idx77(1:n));
lamda7 = vtemp((n+1):(2*n),idx77(1:n));
stemp = lamda7/chi77;
Pmss = stemp';
Kss = stemp*H'/(H*stemp*H' + R);
Ppss = PHI\(Pmss-Gd*Qd*Gd')/PHI';
Pmss = real(Pmss);
Kss = real(Kss);
Ppss = real(Ppss);
clear dtemp idx77 chi77 lamda7 stemp temp77 vtemp qtemp;
//
// Form Augmented Matrices for Performance Evaluation
// ------------------------------------------------
//
Fa = [Ft, -Xt; 0*ones(n,nt), F-X];
Ga = [Gt; 0*ones(n,st)];
GaQaGaT = Ga * Qt * Ga';
na = nt + n;
FFa2 = Fa;
FFa2(na+1,na+1) = 0.;
FFa2D = disc(FFa2,na,delt);
PHIa = split(FFa2D,na);
PGQGPTa = PHIa * GaQaGaT * PHIa';
Qda = (PGQGPTa + GaQaGaT) * 0.5 * delt;
clear FFa2 FFa2D PGQGPTa GaQaGaT Fa Ga na;
Da = [eye(nt), -Dt; 0*ones(n,nt), eye(n)-D];
Ca = [-Ct, C];
Pao = [Pto, 0*ones(nt,n); 0*ones(n,nt), 0*ones(n)];
Aass = [eye(nt), 0*ones(nt,n); Kss*Ht, eye(n)-Kss*H];
Kass = [0*ones(nt,m); Kss];
Pam = Pao;
Pem = Ca * Pam * Ca';
Pm = Pmss;
Pemf = C * Pm * C';
K = Kss;
Pp = Ppss;
Aa = Aass;
Ka = Kass;
Pap = Aa * Pam * Aa' + Ka * Rt * Ka';
Pep = Ca * Pap * Ca';
Pepf = C * Pp * C';
Papc = Da * Pap * Da';
Pepc = Ca * Papc * Ca';
Pepcf = Pepf;
```

```
em = diag(Pem);
ep = diag(Pep);
epc = diag(Pepc);
ETRUE = [em'; ep'; epc'];
emf = diag(Pemf);
epf = diag(Pepf);
epcf = diag(Pepcf);
EFILT = [emf'; epf'; epcf'];
display('Initialization at time to is complete')
display(' ')
// Main Loop:  Iterate for i = 1 to i = ITOTAL
// ---------
itag=1;
xiter=0;
for inum = 1:ITOTAL;...
  Pm = Pmss;...
  Pam = PHIa * Papc * PHIa' + Qda;...
  Pem = Ca * Pam * Ca';...
  Pemf = C * Pm * C';...
  K = Kss;...
  Pp = Ppss;...
  Aa = Aass;...
  Ka = Kass;...
  Pap = Aa * Pam * Aa' + Ka * Rt * Ka';...
  Pep = Ca * Pap * Ca';...
  Pepf = C * Pp * C';...
  Papc = Da * Pap * Da';...
  Pepc = Ca * Papc * Ca';...
  Pepcf = Pepf;...
  em = diag(Pem);...
  ep = diag(Pep);...
  epc = diag(Pepc);...
  ETRUE = [ETRUE; em'; ep'; epc'];...
  emf = diag(Pemf);...
  epf = diag(Pepf);...
  epcf = diag(Pepcf);...
  EFILT = [EFILT; emf'; epf'; epcf'];...
  xiter=xiter+1;...
  xtm=xiter*delt;...
  if xtm=itag;...
    SECONDS=itag,...
    itag=itag+1;...
  end;...
end;
clear xiter xtm itag SECONDS;
//
// End of Main Loop to Conduct Performance Analysis
//****************************************************************
//
// Establish Data Files for Plotting
// --------------------------------
//
RTETRUE = sqrt(ETRUE);
RTEFILT = sqrt(EFILT);
timesT = [0 0 0];
for i = 1:ITOTAL;...
    timesT = [timesT, i, i, i];...
end
times = delt * timesT';
display('Data is now ready for plotting.')
//
//****************************************************************
//
// Generate Plots Iteratively, until user quits
//
MOREPLTS = 1;
```

```
while MOREPLTS >= 1;...
  inquire j 'Enter integer index of variable of interest: ';...
  inquire strtt 'Enter plot start time: ';...
  inquire stpt 'Enter plot stop time: ';...
  stt=((strtt/delt)*3)+1;...
  spt=((stpt/delt)+1)*3;...
  xpl=times(stt:spt);...
  ypl=[RTETRUE(stt:spt,j) RTEFILT(stt:spt,j)];...
  plot(xpl,ypl,'symbol mark 2 4 line xlabel/SECONDS/ ylabel/ERROR/...
    title/LEGEND: TRUE(O), FILTER-COMPUTED(+)/ grid');...
  pause;...
  inquire MOREPLTS 'Do you want more plots  Enter 1 for YES or 0 for NO: ';...
end;
clear j strtt stpt lin77 grd7 stt spt;
erase;
//
// End of Loop to Generate Plots
// ----------------------------
//
clear NEWVARS OKV Pap Pep Pepf Papc Pepc Pepcf em ep epc Ca Kass Qda;
clear emf epf epcf Pem Pemf K Pp Aa Ka MOREPLTS Pao Da Pam PHIa Pm xpl ypl;
//******************************************************************
```

```
//  This Macro plots results from MMAESIM that are thesis
//  quality to be used for printing on the LN03 laser printer.
//
pr1=prb(:,4:9);
pr2=prb(:,10:16);
//
plot(ti,pr1,'strip ...
title/Probability, AOA_Sensor_Failure/ ...
ymin/0/ ymax/1/ ...
xlabel/Time (seconds)/ ...
ylabel/FF|A1|A2|A3|A4|A5/')
hard('pl1s2_hfs1.dat')
//
plot(ti,pr2,'strip ...
title/Probability, AOA_Sensor_Failure/ ...
ymin/0/ ymax/1/ ...
xlabel/Time (seconds)/ ...
ylabel/S1|S2|S3|S4|S5|S6|S7/')
hard('pl2s2_hfs1.dat')
//
//sg1=[st(:,1:8),gs(:,1:2)];
sg1=[st(:,1:4),gs(:,1)];
sg2=[st(:,5:8),gs(:,2)];
//
rtod=57.29578;
sg1(:,1)=rtod*sg1(:,1);
sg1(:,3:4)=rtod*sg1(:,3:4);
sg2(:,1:4)=rtod*sg2(:,1:4);
//
plot(ti,sg1,'strip ...
title/States, AOA_Sensor_Failure/ ...
xlabel/Time (seconds)/ ...
ylabel/theta|u|alpha|q|Ancg/')
hard('pl3s2_hfs1.dat')
//
plot(ti,sg2,'strip ...
title/States, AOA_Sensor_Failure/ ...
xlabel/Time (seconds)/ ...
ylabel/phi|beta|p|r|Aycg/')
hard('pl4s2_hfs1.dat')
//
```

```
//
//        SETUPB1.MXX MATRIXX EXECUTABLE MACRO
//
//              Author:  Capt Gregory L. Stratton
//        Date created:  20 August 1991
//        Date revised:  26 September 1991
//
//  This macro creates and saves to files all the required
//  matrices for a single bank, as used in MMAESIM.  This
//  is normally called from the macro FILECREATE.MXX, however,
//  it can be used by itself as long as the below listed
//  variables currently exist in memory in matrixx.
//
//  Below are listed the required input matrices and variables:
//
//    aa,      The 8x8 plant matrix
// bbcon,      The 8x6 plant B matrix
//     g,      The white Gaussian noise multiplier matrix (as in G*w(t))
//    gd,      An identity matrix of size 14x14
//     q,      The white Gaussian noise covariance matrix
//  fcon,      The 14x14 plant matrix augmented with 1st order actuators
//  bcon,      The 14x6 B matrix of the augmented system
//  hcon,      The 7x14 H matrix of the measurement equation
//  delt,      The sample time (here 1/64 Hz)
//     n,      number of states of the augmented system (here 14)
//  bank,      number of bank of which filters are being created
//  cz14,      column vector of 14 zeros
//   cz8,      column vector of 8 zeros
//  rz14,      row vector of 14 zeros
//
//  Note:  any of the variables above with 'con' in its name may
//         contain rows or columns of zeros, simulating an already
//         detected first failure.
//
//  This macro creates 14 files of the form:  FxxB1.DAT
//  where xx is the filter number (04 thru 17)
//  This macro calls the macro "mtx.mxx" which generates the required
//  matrices for each filter.
//
//
//  S T A R T   M A C R O
//
//
//  fully functional filter, #04
//
fbk=.04+bank/10000;
f=faug;
b=baug;
bb=borig;
h=horig;
exec('mtx.mxx')
fsave 'F04B1.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  also save the fully functional case to the first
//  three 'filters' (truth models) so as to have at
//  least something to start with
//
//  usually the first truth model (f01b1.dat) will
//  be the fully functional case, and f02b1.dat and
//  f03b1.dat will hold the first single and double
//  failure truth models respectively
//
fsave 'f03b1.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
fsave 'f02b1.dat' ...
```

```
fbk aa bb phix bd cqd h gkf r ak akinv detak
fsave 'f01b1.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 1 failure filter, #05
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,9)=cz14;
b(:,1)=cz14;
bb=bbcon;
bb(:,1)=cz8;
exec('mtx.mxx')
fsave 'F05B1.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 2 failure filter, #06
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,10)=cz14;
b(:,2)=cz14;
bb=bbcon;
bb(:,2)=cz8;
exec('mtx.mxx')
fsave 'F06B1.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 3 failure filter, #07
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,11)=cz14;
b(:,3)=cz14;
bb=bbcon;
bb(:,3)=cz8;
exec('mtx.mxx')
fsave 'F07B1.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 4 failure filter, #08
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,12)=cz14;
b(:,4)=cz14;
bb=bbcon;
bb(:,4)=cz8;
exec('mtx.mxx')
fsave 'F08B1.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 5 failure filter, #09
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
```

```
//f(:,13)=cz14;
b(:,5)=cz14;
bb=bbcon;
bb(:,5)=cz8;
exec('mtx.mxx')
fsave 'F09B1.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 1 failure filter, #10
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(1,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F10B1.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 2 failure filter, #11
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(2,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F11B1.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 3 failure filter, #12
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(3,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F12B1.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 4 failure filter, #13
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(4,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F13B1.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 5 failure filter, #14
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(5,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F14B1.dat' ...
```

```
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 6 failure filter, #15
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(6,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F15B1.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 7 failure filter, #16
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(7,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F16B1.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//
//  actuator 6 failure filter, #17
//
//  this has been removed, but can be uncommented out
//  originally, the failed act 6 filter was #10,
//  but is now appended on to the end (as filter #17)
//  if used.
//
//fbk=fbk+.01;
//f=fcon;
//b=bcon;
//h=hcon;
////f(:,14)=cz14;
//b(:,6)=cz14;
//bb=bbcon;
//bb(:,6)=cz8;
//exec('mtx.mxx')
//fsave 'F10B1.dat' ...
//fbk aa bb phix bd cqd h gkf r ak akinv detak
//
return
//
//
//
```

```
//
//        SETUPB2.MXX MATRIXX EXECUTABLE MACRO
//
//              Author:  Capt Gregory L. Stratton
//        Date created:  20 August 1991
//        Date revised:  26 September 1991
//
//  This macro creates and saves to files all the required
//  matrices for a single bank, as used in MMAESIM.  This
//  is normally called from the macro FILECREATE.MXX, however,
//  it can be used by itself as long as the below listed
//  variables currently exist in memory in matrixx.
//
//  Below are listed the required input matrices and variables:
//
//    aa,      The 8x8 plant matrix
// bbcon,      The 8x6 plant B matrix
//     g,      The white Gaussian noise multiplier matrix (as in G*w(t))
//    gd,      An identity matrix of size 14x14
//     q,      The white Gaussian noise covariance matrix
//  fcon,      The 14x14 plant matrix augmented with 1st order actuators
//  bcon,      The 14x6 B matrix of the augmented system
//  hcon,      The 7x14 H matrix of the measurement equation
//  delt,      The sample time (here 1/64 Hz)
//     n,      number of states of the augmented system (here 14)
//  bank,      number of bank of which filters are being created
//  cz14,      column vector of 14 zeros
//   cz8,      column vector of 8 zeros
//  rz14,      row vector of 14 zeros
//
//  Note:  any of the variables above with 'con' in its name may
//         contain rows or columns of zeros, simulating an already
//         detected first failure.
//
//  This macro creates 14 files of the form:  FxxB2.DAT
//  where xx is the filter number (04 thru 17)
//  This macro calls the macro "mtx.mxx" which generates the required
//  matrices for each filter.
//
//
//  S T A R T   M A C R O
//
//
//  fully functional filter, #04
//
fbk=.04+bank/10000;
f=faug;
b=baug;
bb=borig;
h=horig;
exec('mtx.mxx')
fsave 'F04B2.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  also save the fully functional case to the first
//  three 'filters' (truth models) so as to have at
//  least something to start with
//
//  usually the first truth model (f01B2.dat) will
//  be the fully functional case, and f02B2.dat and
//  f03B2.dat will hold the first single and double
//  failure truth models respectively
//
fsave 'f03B2.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
fsave 'f02B2.dat' ...
```

```
fbk aa bb phix bd cqd h gkf r ak akinv detak
fsave 'f01B2.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 1 failure filter, #05
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,9)=cz14;
b(:,1)=cz14;
bb=bbcon;
bb(:,1)=cz8;
exec('mtx.mxx')
fsave 'F05B2.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 2 failure filter, #06
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,10)=cz14;
b(:,2)=cz14;
bb=bbcon;
bb(:,2)=cz8;
exec('mtx.mxx')
fsave 'F06B2.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 3 failure filter, #07
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,11)=cz14;
b(:,3)=cz14;
bb=bbcon;
bb(:,3)=cz8;
exec('mtx.mxx')
fsave 'F07B2.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 4 failure filter, #08
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,12)=cz14;
b(:,4)=cz14;
bb=bbcon;
bb(:,4)=cz8;
exec('mtx.mxx')
fsave 'F08B2.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 5 failure filter, #09
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
```

```
//f(:,13)=cz14;
b(:,5)=cz14;
bb=bbcon;
bb(:,5)=cz8;
exec('mtx.mxx')
fsave 'F09B2.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 1 failure filter, #10
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(1,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F10B2.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 2 failure filter, #11
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(2,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F11B2.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 3 failure filter, #12
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(3,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F12B2.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 4 failure filter, #13
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(4,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F13B2.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 5 failure filter, #14
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(5,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F14B2.dat' ...
```

```
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 6 failure filter, #15
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(6,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F15B2.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 7 failure filter, #16
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(7,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F16B2.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//
//  actuator 6 failure filter, #17
//
//  this has been removed, but can be uncommented out
//  originally, the failed act 6 filter was #10,
//  but is now appended on to the end (as filter #17)
//  if used.
//
//fbk=fbk+.01;
//f=fcon;
//b=bcon;
//h=hcon;
////f(:,14)=cz14;
//b(:,6)=cz14;
//bb=bbcon;
//bb(:,6)=cz8;
//exec('mtx.mxx')
//fsave 'F10B2.dat' ...
//fbk aa bb phix bd cqd h gkf r ak akinv detak
//
return
//
//
//
```

```
//
//        SETUPB3.MXX MATRIXX EXECUTABLE MACRO
//
//              Author:  Capt Gregory L. Stratton
//        Date created:  20 August 1991
//        Date revised:  26 September 1991
//
//  This macro creates and saves to files all the required
//  matrices for a single bank, as used in MMAESIM.  This
//  is normally called from the macro FILECREATE.MXX, however,
//  it can be used by itself as long as the below listed
//  variables currently exist in memory in matrixx.
//
//  Below are listed the required input matrices and variables:
//
//    aa,      The 8x8 plant matrix
// bbcon,      The 8x6 plant B matrix
//     g,      The white Gaussian noise multiplier matrix (as in G*w(t))
//    gd,      An identity matrix of size 14x14
//     q,      The white Gaussian noise covariance matrix
//  fcon,      The 14x14 plant matrix augmented with 1st order actuators
//  bcon,      The 14x6 B matrix of the augmented system
//  hcon,      The 7x14 H matrix of the measurement equation
//  delt,      The sample time (here 1/64 Hz)
//     n,      number of states of the augmented system (here 14)
//  bank,      number of bank of which filters are being created
//  cz14,      column vector of 14 zeros
//   cz8,      column vector of 8 zeros
//  rz14,      row vector of 14 zeros
//
//  Note:  any of the variables above with 'con' in its name may
//         contain rows or columns of zeros, simulating an already
//         detected first failure.
//
//  This macro creates 14 files of the form:  FxxB3.DAT
//  where xx is the filter number (04 thru 17)
//  This macro calls the macro "mtx.mxx" which generates the required
//  matrices for each filter.
//
//
//  S T A R T   M A C R O
//
//
//  fully functional filter, #04
//
fbk=.04+bank/10000;
f=faug;
b=baug;
bb=borig;
h=horig;
exec('mtx.mxx')
fsave 'F04B3.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  also save the fully functional case to the first
//  three 'filters' (truth models) so as to have at
//  least something to start with
//
//  usually the first truth model (f01B3.dat) will
//  be the fully functional case, and f02B3.dat and
//  f03B3.dat will hold the first single and double
//  failure truth models respectively
//
fsave 'f03B3.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
fsave 'f02B3.dat' ...
```

```
fbk aa bb phix bd cqd h gkf r ak akinv detak
fsave 'f01B3.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 1 failure filter, #05
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,9)=cz14;
b(:,1)=cz14;
bb=bbcon;
bb(:,1)=cz8;
exec('mtx.mxx')
fsave 'F05B3.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 2 failure filter, #06
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,10)=cz14;
b(:,2)=cz14;
bb=bbcon;
bb(:,2)=cz8;
exec('mtx.mxx')
fsave 'F06B3.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 3 failure filter, #07
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,11)=cz14;
b(:,3)=cz14;
bb=bbcon;
bb(:,3)=cz8;
exec('mtx.mxx')
fsave 'F07B3.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 4 failure filter, #08
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,12)=cz14;
b(:,4)=cz14;
bb=bbcon;
bb(:,4)=cz8;
exec('mtx.mxx')
fsave 'F08B3.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 5 failure filter, #09
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
```

```
//f(:,13)=cz14;
b(:,5)=cz14;
bb=bbcon;
bb(:,5)=cz8;
exec('mtx.mxx')
fsave 'F09B3.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 1 failure filter, #10
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(1,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F10B3.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 2 failure filter, #11
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(2,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F11B3.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 3 failure filter, #12
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(3,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F12B3.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detek
//
//  sensor 4 failure filter, #13
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(4,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F13B3.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 5 failure filter, #14
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(5,:)=rz14;
bb=bbcon;
axec('mtx.mxx')
fsave 'F14B3.dat' ...
```

```
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 6 failure filter, #15
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(6,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F15B3.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 7 failure filter, #16
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(7,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F16B3.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//
//  actuator 6 failure filter, #17
//
//  this has been removed, but can be uncommented out
//  originally, the failed act 6 filter was #10,
//  but is now appended on to the end (as filter #17)
//  if used.
//
//fbk=fbk+.01;
//f=fcon;
//b=bcon;
//h=hcon;
////f(:,14)=cz14;
//b(:,6)=cz14;
//bb=bbcon;
//bb(:,6)=cz8;
//exec('mtx.mxx')
//fsave 'F10B3.dat' ...
//fbk aa bb phix bd cqd h gkf r ak akinv detak
//
return
//
//
//^
```

```
//
//         SETUPB4.MXX MATRIXX EXECUTABLE MACRO
//
//                Author:  Capt Gregory L. Stratton
//          Date created:  20 August 1991
//          Date revised:  26 September 1991
//
//  This macro creates and saves to files all the required
//  matrices for a single bank, as used in MMAESIM.  This
//  is normally called from the macro FILECREATE.MXX, however,
//  it can be used by itself as long as the below listed
//  variables currently exist in memory in matrixx.
//
//  Below are listed the required input matrices and variables:
//
//     aa,      The 8x8 plant matrix
// bbcon,       The 8x6 plant B matrix
//     g,       The white Gaussian noise multiplier matrix (as in G*w(t))
//    gd,       An identity matrix of size 14x14
//     q,       The white Gaussian noise covariance matrix
//  fcon,       The 14x14 plant matrix augmented with 1st order actuators
//  bcon,       The 14x6 B matrix of the augmented system
//  hcon,       The 7x14 H matrix of the measurement equation
//  delt,       The sample time (here 1/64 Hz)
//     n,       number of states of the augmented system (here 14)
//  bank,       number of bank of which filters are being created
//  cz14,       column vector of 14 zeros
//   cz8,       column vector of 8 zeros
//  rz14,       row vector of 14 zeros
//
//  Note:  any of the variables above with 'con' in its name may
//         contain rows or columns of zeros, simulating an already
//         detected first failure.
//
//  This macro creates 14 files of the form:  FxxB4.DAT
//  where xx is the filter number (04 thru 17)
//  This macro calls the macro "mtx.mxx" which generates the required
//  matrices for each filter.
//
//
//  S T A R T   M A C R O
//
//
//  fully functional filter, #04
//
fbk=.04+bank/10000;
f=faug;
b=baug;
bb=borig;
h=horig;
exec('mtx.mxx')
fsave 'F04B4.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  also save the fully functional case to the first
//  three 'filters' (truth models) so as to have at
//  least something to start with
//
//  usually the first truth model (f01B4.dat) will
//  be the fully functional case, and f02B4.dat and
//  f03B4.dat will hold the first single and double
//  failure truth models respectively
//
fsave 'f03B4.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
fsave 'f02B4.dat' ...
```

```
fbk aa bb phix bd cqd h gkf r ak akinv detak
fsave 'f01B4.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 1 failure filter, #05
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,9)=cz14;
b(:,1)=cz14;
bb=bbcon;
bb(:,1)=cz8;
exec('mtx.mxx')
fsave 'F05B4.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 2 failure filter, #06
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,10)=cz14;
b(:,2)=cz14;
bb=bbcon;
bb(:,2)=cz8;
exec('mtx.mxx')
fsave 'F06B4.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 3 failure filter, #07
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,11)=cz14;
b(:,3)=cz14;
bb=bbcon;
bb(:,3)=cz8;
exec('mtx.mxx')
fsave 'F07B4.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 4 failure filter, #08
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,12)=cz14;
b(:,4)=cz14;
bb=bbcon;
bb(:,4)=cz8;
exec('mtx.mxx')
fsave 'F08B4.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 5 failure filter, #09
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
```

```
//f(:,13)=cz14;
b(:,5)=cz14;
bb=bbcon;
bb(:,5)=cz8;
exec('mtx.mxx')
fsave 'F09B4.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 1 failure filter, #10
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(1,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F10B4.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 2 failure filter, #11
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(2,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F11B4.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 3 failure filter, #12
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(3,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F12B4.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 4 failure filter, #13
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(4,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F13B4.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 5 failure filter, #14
//
fbk=fbk+.01;
f=*con;
b=bcon;
h=hcon;
h(5,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F14B4.dat' ...
```

```
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 6 failure filter, #15
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(6,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F15B4.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 7 failure filter, #16
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(7,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F16B4.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//
//  actuator 6 failure filter, #17
//
//  this has been removed, but can be uncommented out
//  originally, the failed act 6 filter was #10,
//  but is now appended on to the end (as filter #17)
//  if used.
//
//fbk=fbk+.01;
//f=fcon;
//b=bcon;
//h=hcon;
////f(:,14)=cz14;
//b(:,6)=cz14;
//bb=bbcon;
//bb(:,6)=cz8;
//exec('mtx.mxx')
//fsave 'F10B4.dat' ...
//fbk aa bb phix bd cqd h gkf r ak akinv detak
//
return
//
//
//↑
```

```
//
//        SETUPB5.MXX MATRIXX EXECUTABLE MACRO
//
//              Author:  Capt Gregory L. Stratton
//        Date created:  20 August 1991
//        Date revised:  26 September 1991
//
//  This macro creates and saves to files all the required
//  matrices for a single bank, as used in MMAESIM.  This
//  is normally called from the macro FILECREATE.MXX, however,
//  it can be used by itself as long as the below listed
//  variables currently exist in memory in matrixx.
//
//  Below are listed the required input matrices and variables:
//
//     aa,      The 8x8 plant matrix
// bbcon,       The 8x6 plant B matrix
//      g,      The white Gaussian noise multiplier matrix (as in G*w(t))
//     gd,      An identity matrix of size 14x14
//      q,      The white Gaussian noise covariance matrix
//  fcon,       The 14x14 plant matrix augmented with 1st order actuators
//  bcon,       The 14x6 B matrix of the augmented system
//  hcon,       The 7x14 H matrix of the measurement equation
//  delt,       The sample time (here 1/64 Hz)
//     n,       number of states of the augmented system (here 14)
//  bank,       number of bank of which filters are being created
//  cz14,       column vector of 14 zeros
//   cz8,       column vector of 8 zeros
//  rz14,       row vector of 14 zeros
//
//  Note:  any of the variables above with 'con' in its name may
//         contain rows or columns of zeros, simulating an already
//         detected first failure.
//
//  This macro creates 14 files of the form:  FxxB5.DAT
//  where xx is the filter number (04 thru 17)
//  This macro calls the macro "mtx.mxx" which generates the required
//  matrices for each filter.
//
//
//  S T A R T   M A C R O
//
//
//  fully functional filter, #04
//
fbk=.04+bank/10000;
f=faug;
b=baug;
bb=borig;
h=horig;
exec('mtx.mxx')
fsave 'F04B5.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  also save the fully functional case to the first
//  three 'filters' (truth models) so as to have at
//  least something to start with
//
//  usually the first truth model (f01B5.dat) will
//  be the fully functional case, and f02B5.dat and
//  f03B5.dat will hold the first single and double
//  failure truth models respectively
//
fsave 'f03B5.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
fsave 'f02B5.dat' ...
```

```
fbk aa bb phix bd cqd h gkf r ak akinv detak
fsave 'f01B5.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 1 failure filter, #05
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,9)=cz14;
b(:,1)=cz14;
bb=bbcon;
bb(:,1)=cz8;
exec('mtx.mxx')
fsave 'F05B5.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 2 failure filter, #06
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,10)=cz14;
b(:,2)=cz14;
bb=bbcon;
bb(:,2)=cz8;
exec('mtx.mxx')
fsave 'F06B5.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 3 failure filter, #07
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,11)=cz14;
b(:,3)=cz14;
bb=bbcon;
bb(:,3)=cz8;
exec('mtx.mxx')
fsave 'F07B5.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 4 failure filter, #08
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,12)=cz14;
b(:,4)=cz14;
bb=bbcon;
bb(:,4)=cz8;
exec('mtx.mxx')
fsave 'F08B5.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 5 failure filter, #09
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
```

```
//f(:,13)=cz14;
b(:,5)=cz14;
bb=bbcon;
bb(:,5)=cz8;
exec('mtx.mxx')
fsave 'F09B5.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 1 failure filter, #10
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(1,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F10B5.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 2 failure filter, #11
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(2,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F11B5.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 3 failure filter, #12
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(3,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F12B5.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 4 failure filter, #13
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(4,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F13B5.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 5 failure filter, #14
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(5,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F14B5.dat' ...
```

```
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 6 failure filter, #15
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(6,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F15B5.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 7 failure filter, #16
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(7,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F16B5.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//
//  actuator 6 failure filter, #17
//
//  this has been removed, but can be uncommented out
//  originally, the failed act 6 filter was #10,
//  but is now appended on to the end (as filter #17)
//  if used.
//
//fbk=fbk+.01;
//f=fcon;
//b=bcon;
//h=hcon;
////f(:,14)=cz14;
//b(:,6)=cz14;
//bb=bbcon;
//bb(:,6)=cz8;
//exec('mtx.mxx')
//fsave 'F10B5.dat' ...
//fbk aa bb phix bd cqd h gkf r ak akinv detak
//
return
//
//
//
```

```
//
//        SETUPB6.MXX MATRIXX EXECUTABLE MACRO
//
//              Author:  Capt Gregory L. Stratton
//        Date created:  20 August 1991
//        Date revised:  26 September 1991
//
//  This macro creates and saves to files all the required
//  matrices for a single bank, as used in MMAESIM.  This
//  is normally called from the macro FILECREATE.MXX, however,
//  it can be used by itself as long as the below listed
//  variables currently exist in memory in matrixx.
//
//  Below are listed the required input matrices and variables:
//
//    aa,     The 8x8 plant matrix
// bbcon,     The 8x6 plant B matrix
//     g,     The white Gaussian noise multiplier matrix (as in G*w(t))
//    gd,     An identity matrix of size 14x14
//     q,     The white Gaussian noise covariance matrix
//  fcon,     The 14x14 plant matrix augmented with 1st order actuators
//  bcon,     The 14x6 B matrix of the augmented system
//  hcon,     The 7x14 H matrix of the measurement equation
//  delt,     The sample time (here 1/64 Hz)
//     n,     number of states of the augmented system (here 14)
//  bank,     number of bank of which filters are being created
//  cz14,     column vector of 14 zeros
//   cz8,     column vector of 8 zeros
//  rz14,     row vector of 14 zeros
//
//  Note:  any of the variables above with 'con' in its name may
//         contain rows or columns of zeros, simulating an already
//         detected first failure.
//
//  This macro creates 14 files of the form:  FxxB6.DAT
//  where xx is the filter number (04 thru 17)
//  This macro calls the macro "mtx.mxx" which generates the required
//  matrices for each filter.
//
//
//  S T A R T   M A C R O
//
//
//  fully functional filter, #04
//
fbk=.04+bank/10000;
f=faug;
b=baug;
bb=borig;
h=horig;
exec('mtx.mxx')
fsave 'F04B6.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  also save the fully functional case to the first
//  three 'filters' (truth models) so as to have at
//  least something to start with
//
//  usually the first truth model (f01B6.dat) will
//  be the fully functional case, and f02B6.dat and
//  f03B6.dat will hold the first single and double
//  failure truth models respectively
//
fsave 'f03B6.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
fsave 'f02B6.dat' ...
```

```
fbk aa bb phix bd cqd h gkf r ak akinv detak
fsave 'f01B6.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 1 failure filter, #05
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,9)=cz14;
b(:,1)=cz14;
bb=bbcon;
bb(:,1)=cz8;
exec('mtx.mxx')
fsave 'F05B6.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 2 failure filter, #06
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,10)=cz14;
b(:,2)=cz14;
bb=bbcon;
bb(:,2)=cz8;
exec('mtx.mxx')
fsave 'F06B6.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 3 failure filter, #07
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,11)=cz14;
b(:,3)=cz14;
bb=bbcon;
bb(:,3)=cz8;
exec('mtx.mxx')
fsave 'F07B6.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 4 failure filter, #08
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,12)=cz14;
b(:,4)=cz14;
bb=bbcon;
bb(:,4)=cz8;
exec('mtx.mxx')
fsave 'F08B6.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 5 failure filter, #09
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
```

```
//f(:,13)=cz14;
b(:,5)=cz14;
bb=bbcon;
bb(:,5)=cz8;
exec('mtx.mxx')
fsave 'F09B6.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 1 failure filter, #10
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(1,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F10B6.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 2 failure filter, #11
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(2,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F11B6.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 3 failure filter, #12
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(3,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F12B6.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 4 failure filter, #13
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(4,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F13B6.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 5 failure filter, #14
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(5,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F14B6.dat' ...
```

```
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 6 failure filter, #15
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(6,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F15B6.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 7 failure filter, #16
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(7,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F16B6.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//
//  actuator 6 failure filter, #17
//
//  this has been removed, but can be uncommented out
//  originally, the failed act 6 filter was #10,
//  but is now appended on to the end (as filter #17)
//  if used.
//
//fbk=fbk+.01;
//f=fcon;
//b=bcon;
//h=hcon;
////f(:,14)=cz14;
//b(:,6)=cz14;
//bb=bbcon;
//bb(:,6)=cz8;
//exec('mtx.mxx')
//fsave 'F10B6.dat' ...
//fbk aa bb phix bd cqd h gkf r ak akinv detak
//
return
//
//
//
```

```
//
//        SETUPB7.MXX MATRIXX EXECUTABLE MACRO
//
//              Author:  Capt Gregory L. Stratton
//        Date created:  20 August 1991
//        Date revised:  26 September 1991
//
//  This macro creates and saves to files all the required
//  matrices for a single bank, as used in MMAESIM.  This
//  is normally called from the macro FILECREATE.MXX, however,
//  it can be used by itself as long as the below listed
//  variables currently exist in memory in matrixx.
//
//  Below are listed the required input matrices and variables:
//
//    aa,      The 8x8 plant matrix
// bbcon,      The 8x6 plant B matrix
//     g,      The white Gaussian noise multiplier matrix (as in G*w(t))
//    gd,      An identity matrix of size 14x14
//     q,      The white Gaussian noise covariance matrix
//  fcon,      The 14x14 plant matrix augmented with 1st order actuators
//  bcon,      The 14x6 B matrix of the augmented system
//  hcon,      The 7x14 H matrix of the measurement equation
//  delt,      The sample time (here 1/64 Hz)
//     n,      number of states of the augmented system (here 14)
//  bank,      number of bank of which filters are being created
//  cz14,      column vector of 14 zeros
//   cz8,      column vector of 8 zeros
//  rz14,      row vector of 14 zeros
//
//  Note:  any of the variables above with 'con' in its name may
//         contain rows or columns of zeros, simulating an already
//         detected first failure.
//
//  This macro creates 14 files of the form:  FxxB7.DAT
//  where xx is the filter number (04 thru 17)
//  This macro calls the macro "mtx.mxx" which generates the required
//  matrices for each filter.
//
//
//  S T A R T   M A C R O
//
//
//  fully functional filter, #04
//
fbk=.04+bank/10000;
f=faug;
b=baug;
bb=borig;
h=horig;
exec('mtx.mxx')
fsave 'F04B7.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  also save the fully functional case to the first
//  three 'filters' (truth models) so as to have at
//  least something to start with
//
//  usually the first truth model (f01B7.dat) will
//  be the fully functional case, and f02B7.dat and
//  f03B7.dat will hold the first single and double
//  failure truth models respectively
//
fsave 'f03B7.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
fsave 'f02B7.dat' ...
```

```
fbk aa bb phix bd cqd h gkf r ak akinv detak
fsave 'f01B7.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 1 failure filter, #05
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,9)=cz14;
b(:,1)=cz14;
bb=bbcon;
bb(:,1)=cz8;
exec('mtx.mxx')
fsave 'F05B7.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 2 failure filter, #06
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,10)=cz14;
b(:,2)=cz14;
bb=bbcon;
bb(:,2)=cz8;
exec('mtx.mxx')
fsave 'F06B7.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 3 failure filter, #07
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,11)=cz14;
b(:,3)=cz14;
bb=bbcon;
bb(:,3)=cz8;
exec('mtx.mxx')
fsave 'F07B7.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 4 failure filter, #08
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,12)=cz14;
b(:,4)=cz14;
bb=bbcon;
bb(:,4)=cz8;
exec('mtx.mxx')
fsave 'F08B7.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 5 failure filter, #09
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
```

```
//f(:,13)=cz14;
b(:,5)=cz14;
bb=bbcon;
bb(:,5)=cz8;
exec('mtx.mxx')
fsave 'F09B7.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 1 failure filter, #10
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(1,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F10B7.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 2 failure filter, #11
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(2,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F11B7.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 3 failure filter, #12
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(3,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F12B7.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 4 failure filter, #13
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(4,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F13B7.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 5 failure filter, #14
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(5,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F14B7.dat' ...
```

```
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 6 failure filter, #15
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(6,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F15B7.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 7 failure filter, #16
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(7,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F16B7.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//
//  actuator 6 failure filter, #17
//
//  this has been removed, but can be uncommented out
//  originally, the failed act 6 filter was #10,
//  but is now appended on to the end (as filter #17)
//  if used.
//
//fbk=fbk+.01;
//f=fcon;
//b=bcon;
//h=hcon;
////f(:,14)=cz14;
//b(:,6)=cz14;
//bb=bbcon;
//bb(:,6)=cz8;
//exec('mtx.mxx')
//fsave 'F10B7.dat' ...
//fbk aa bb phix bd cqd h gkf r ak akinv detak
//
return
//
//
//
```

```
//
//        SETUPB8.MXX MATRIXX EXECUTABLE MACRO
//
//              Author:  Capt Gregory L. Stratton
//        Date created:  20 August 1991
//        Date revised:  26 September 1991
//
//  This macro creates and saves to files all the required
//  matrices for a single bank, as used in MMAESIM.  This
//  is normally called from the macro FILECREATE.MXX, however,
//  it can be used by itself as long as the below listed
//  variables currently exist in memory in matrixx.
//
//  Below are listed the required input matrices and variables:
//
//    aa,     The 8x8 plant matrix
// bbcon,     The 8x6 plant B matrix
//     g,     The white Gaussian noise multiplier matrix (as in G*w(t))
//    gd,     An identity matrix of size 14x14
//     q,     The white Gaussian noise covariance matrix
//  fcon,     The 14x14 plant matrix augmented with 1st order actuators
//  bcon,     The 14x6 B matrix of the augmented system
//  hcon,     The 7x14 H matrix of the measurement equation
//  delt,     The sample time (here 1/64 Hz)
//     n,     number of states of the augmented system (here 14)
//  bank,     number of bank of which filters are being created
//  cz14,     column vector of 14 zeros
//   cz8,     column vector of 8 zeros
//  rz14,     row vector of 14 zeros
//
//  Note:  any of the variables above with 'con' in its name may
//         contain rows or columns of zeros, simulating an already
//         detected first failure.
//
//  This macro creates 14 files of the form:  FxxB8.DAT
//  where xx is the filter number (04 thru 17)
//  This macro calls the macro "mtx.mxx" which generates the required
//  matrices for each filter.
//
//
//  S T A R T   M A C R O
//
//
//  fully functional filter, #04
//
fbk=.04+bank/10000;
f=faug;
b=baug;
bb=borig;
h=horig;
exec('mtx.mxx')
fsave 'F04B8.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  also save the fully functional case to the first
//  three 'filters' (truth models) so as to have at
//  least something to start with
//
//  usually the first truth model (f01B8.dat) will
//  be the fully functional case, and f02B8.dat and
//  f03B8.dat will hold the first single and double
//  failure truth models respectively
//
fsave 'f03B8.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
fsave 'f02B8.dat' ...
```

```
fbk aa bb phix bd cqd h gkf r ak akinv detak
fsave 'f01B8.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 1 failure filter, #05
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,9)=cz14;
b(:,1)=cz14;
bb=bbcon;
bb(:,1)=cz8;
exec('mtx.mxx')
fsave 'F05B8.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 2 failure filter, #06
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,10)=cz14;
b(:,2)=cz14;
bb=bbcon;
bb(:,2)=cz8;
exec('mtx.mxx')
fsave 'F06B8.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 3 failure filter, #07
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,11)=cz14;
b(:,3)=cz14;
bb=bbcon;
bb(:,3)=cz8;
exec('mtx.mxx')
fsave 'F07B8.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 4 failure filter, #08
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,12)=cz14;
b(:,4)=cz14;
bb=bbcon;
bb(:,4)=cz8;
exec('mtx.mxx')
fsave 'F08B8.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 5 failure filter, #09
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
```

```
//f(:,13)=cz14;
b(:,5)=cz14;
bb=bbcon;
bb(:,5)=cz8;
exec('mtx.mxx')
fsave 'F09B8.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 1 failure filter, #10
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(1,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F10B8.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 2 failure filter, #11
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(2,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F11B8.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 3 failure filter, #12
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(3,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F12B8.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 4 failure filter, #13
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(4,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F13B8.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 5 failure filter, #14
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(5,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F14B8.dat' ...
```

```
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 6 failure filter, #15
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(6,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F15B8.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 7 failure filter, #16
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(7,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F16B8.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//
//  actuator 6 failure filter, #17
//
//  this has been removed, but can be uncommented out
//  originally, the failed act 6 filter was #10,
//  but is now appended on to the end (as filter #17)
//  if used.
//
//fbk=fbk+.01;
//f=fcon;
//b=bcon;
//h=hcon;
////f(:,14)=cz14;
//b(:,6)=cz14;
//bb=bbcon;
//bb(:,6)=cz8;
//exec('mtx.mxx')
//fsave 'F10B8.dat' ...
//fbk aa bb phix bd cqd h gkf r ak akinv detak
//
return
//
//
//
```

```
//
//        SETUPB9.MXX MATRIXX EXECUTABLE MACRO
//
//              Author:  Capt Gregory L. Stratton
//        Date created:  20 August 1991
//        Date revised:  26 September 1991
//
//  This macro creates and saves to files all the required
//  matrices for a single bank, as used in MMAESIM.  This
//  is normally called from the macro FILECREATE.MXX, however,
//  it can be used by itself as long as the below listed
//  variables currently exist in memory in matrixx.
//
//  Below are listed the required input matrices and variables:
//
//     aa,      The 8x8 plant matrix
// bbcon,       The 8x6 plant B matrix
//      g,      The white Gaussian noise multiplier matrix (as in G*w(t))
//     gd,      An identity matrix of size 14x14
//      q,      The white Gaussian noise covariance matrix
//  fcon,       The 14x14 plant matrix augmented with 1st order actuators
//  bcon,       The 14x6 B matrix of the augmented system
//  hcon,       The 7x14 H matrix of the measurement equation
//  delt,       The sample time (here 1/64 Hz)
//      n,      number of states of the augmented system (here 14)
//  bank,       number of bank of which filters are being created
//  cz14,       column vector of 14 zeros
//   cz8,       column vector of 8 zeros
//  rz14,       row vector of 14 zeros
//
//  Note:  any of the variables above with 'con' in its name may
//         contain rows or columns of zeros, simulating an already
//         detected first failure.
//
//  This macro creates 14 files of the form:  FxxB9.DAT
//  where xx is the filter number (04 thru 17)
//  This macro calls the macro "mtx.mxx" which generates the required
//  matrices for each filter.
//
//
//  S T A R T   M A C R O
//
//
//  fully functional filter, #04
//
fbk=.04+bank/10000;
f=faug;
b=baug;
bb=borig;
h=horig;
exec('mtx.mxx')
fsave 'F04B9.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  also save the fully functional case to the first
//  three 'filters' (truth models) so as to have at
//  least something to start with
//
//  usually the first truth model (f01B9.dat) will
//  be the fully functional case, and f02B9.dat and
//  f03B9.dat will hold the first single and double
//  failure truth models respectively
//
fsave 'f03B9.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
fsave 'f02B9.dat' ...
```

```
fbk aa bb phix bd cqd h gkf r ak akinv detak
fsave 'f01B9.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 1 failure filter, #05
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,9)=cz14;
b(:,1)=cz14;
bb=bbcon;
bb(:,1)=cz8;
exec('mtx.mxx')
fsave 'F05B9.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 2 failure filter, #06
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,10)=cz14;
b(:,2)=cz14;
bb=bbcon;
bb(:,2)=cz8;
exec('mtx.mxx')
fsave 'F06B9.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 3 failure filter, #07
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,11)=cz14;
b(:,3)=cz14;
bb=bbcon;
bb(:,3)=cz8;
exec('mtx.mxx')
fsave 'F07B9.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 4 failure filter, #08
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,12)=cz14;
b(:,4)=cz14;
bb=bbcon;
bb(:,4)=cz8;
exec('mtx.mxx')
fsave 'F08B9.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 5 failure filter, #09
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
```

```
//f(:,13)=cz14;
b(:,5)=cz14;
bb=bbcon;
bb(:,5)=cz8;
exec('mtx.mxx')
fsave 'F09B9.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 1 failure filter, #10
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(1,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F10B9.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 2 failure filter, #11
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(2,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F11B9.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 3 failure filter, #12
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(3,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F12B9.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 4 failure filter, #13
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(4,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F13B9.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 5 failure filter, #14
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(5,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F14B9.dat' ...
```

```
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 6 failure filter, #15
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(6,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F15B9.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 7 failure filter, #16
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(7,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F16B9.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//
//  actuator 6 failure filter, #17
//
//  this has been removed, but can be uncommented out
//  originally, the failed act 6 filter was #10,
//  but is now appended on to the end (as filter #17)
//  if used.
//
//fbk=fbk+.01;
//f=fcon;
//b=bcon;
//h=hcon;
////f(:,14)=cz14;
//b(:,6)=cz14;
//bb=bbcon;
//bb(:,6)=cz8;
//exec('mtx.mxx')
//fsave 'F10B9.dat' ...
//fbk aa bb phix bd cqd h gkf r ak akinv detak
//
return
//
//
//
```

```
//
//        SETUPX0.MXX MATRIXX EXECUTABLE MACRO
//
//              Author:  Capt Gregory L. Stratton
//        Date created:  20 August 1991
//        Date revised:  26 September 1991
//
// This macro creates and saves to files all the required
// matrices for a single bank, as used in MMAESIM.  This
// is normally called from the macro FILECREATE.MXX, however,
// it can be used by itself as long as the below listed
// variables currently exist in memory in matrixx.
//
// Below are listed the required input matrices and variables:
//
//    aa,     The 8x8 plant matrix
// bbcon,     The 8x6 plant B matrix
//     g,     The white Gaussian noise multiplier matrix (as in G*w(t))
//    gd,     An identity matrix of size 14x14
//     q,     The white Gaussian noise covariance matrix
//  fcon,     The 14x14 plant matrix augmented with 1st order actuators
//  bcon,     The 14x6 B matrix of the augmented system
//  hcon,     The 7x14 H matrix of the measurement equation
//  delt,     The sample time (here 1/64 Hz)
//     n,     number of states of the augmented system (here 14)
//  bank,     number of bank of which filters are being created
//  cz14,     column vector of 14 zeros
//   cz8,     column vector of 8 zeros
//  rz14,     row vector of 14 zeros
//
// Note:  any of the variables above with 'con' in its name may
//        contain rows or columns of zeros, simulating an already
//        detected first failure.
//
// This macro creates 14 files of the form:  FxxX0.DAT
// where xx is the filter number (04 thru 17)
// This macro calls the macro "mtx.mxx" which generates the required
// matrices for each filter.
//
//
// S T A R T   M A C R O
//
//
// fully functional filter, #04
//
fbk=.04+bank/10000;
f=faug;
b=baug;
bb=borig;
h=horig;
exec('mtx.mxx')
fsave 'F04X0.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
// also save the fully functional case to the first
// three 'filters' (truth models) so as to have at
// least something to start with
//
// usually the first truth model (f01X0.dat) will
// be the fully functional case, and f02X0.dat and
// f03X0.dat will hold the first single and double
// failure truth models respectively
//
fsave 'f03X0.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
fsave 'f02X0.dat' ...
```

```
fbk aa bb phix bd cqd h gkf r ak akinv detak
fsave 'f01X0.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 1 failure filter, #05
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,9)=cz14;
b(:,1)=cz14;
bb=bbcon;
bb(:,1)=cz8;
exec('mtx.mxx')
fsave 'F05X0.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 2 failure filter, #06
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,10)=cz14;
b(:,2)=cz14;
bb=bbcon;
bb(:,2)=cz8;
exec('mtx.mxx')
fsave 'F06X0.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 3 failure filter, #07
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,11)=cz14;
b(:,3)=cz14;
bb=bbcon;
bb(:,3)=cz8;
exec('mtx.mxx')
fsave 'F07X0.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 4 failure filter, #08
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,12)=cz14;
b(:,4)=cz14;
bb=bbcon;
bb(:,4)=cz8;
exec('mtx.mxx')
fsave 'F08X0.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 5 failure filter, #09
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
```

```
//f(:,13)=cz14;
b(:,5)=cz14;
bb=bbcon;
bb(:,5)=cz8;
exec('mtx.mxx')
fsave 'F09X0.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 1 failure filter, #10
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(1,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F10X0.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 2 failure filter, #11
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(2,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F11X0.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 3 failure filter, #12
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(3,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F12X0.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 4 failure filter, #13
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(4,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F13X0.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 5 failure filter, #14
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(5,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F14X0.dat' ...
```

```
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 6 failure filter, #15
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(6,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F15X0.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 7 failure filter, #16
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(7,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F16X0.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//
//  actuator 6 failure filter, #17
//
//  this has been removed, but can be uncommented out
//  originally, the failed act 6 filter was #10,
//  but is now appended on to the end (as filter #17)
//  if used.
//
//fbk=fbk+.01;
//f=fcon;
//b=bcon;
//h=hcon;
////f(:,14)=cz14;
//b(:,6)=cz14;
//bb=bbcon;
//bb(:,6)=cz8;
//exec('mtx.mxx')
//fsave 'F10X0.dat' ...
//fbk aa bb phix bd cqd h gkf r ak akinv detak
//
return
//
//
//
```

```
//
//        SETUPX1.MXX MATRIXX EXECUTABLE MACRO
//
//              Author:  Capt Gregory L. Stratton
//        Date created:  20 August 1991
//        Date revised:  26 September 1991
//
//  This macro creates and saves to files all the required
//  matrices for a single bank, as used in MMAESIM.  This
//  is normally called from the macro FILECREATE.MXX, however,
//  it can be used by itself as long as the below listed
//  variables currently exist in memory in matrixx.
//
//  Below are listed the required input matrices and variables:
//
//    aa,      The 8x8 plant matrix
// bbcon,      The 8x6 plant B matrix
//     g,      The white Gaussian noise multiplier matrix (as in G*w(t))
//    gd,      An identity matrix of size 14x14
//     q,      The white Gaussian noise covariance matrix
//  fcon,      The 14x14 plant matrix augmented with 1st order actuators
//  bcon,      The 14x6 B matrix of the augmented system
//  hcon,      The 7x14 H matrix of the measurement equation
//  delt,      The sample time (here 1/64 Hz)
//     n,      number of states of the augmented system (here 14)
//  bank,      number of bank of which filters are being created
//  cz14,      column vector of 14 zeros
//   cz8,      column vector of 8 zeros
//  rz14,      row vector of 14 zeros
//
//  Note:  any of the variables above with 'con' in its name may
//         contain rows or columns of zeros, simulating an already
//         detected first failure.
//
//  This macro creates 14 files of the form:  FxxX1.DAT
//  where xx is the filter number (04 thru 17)
//  This macro calls the macro "mtx.mxx" which generates the required
//  matrices for each filter.
//
//
//  S T A R T   M A C R O
//
//
//  fully functional filter, #04
//
fbk=.04+bank/10000;
f=faug;
b=baug;
bb=borig;
h=horig;
exec('mtx.mxx')
fsave 'F04X1.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  also save the fully functional case to the first
//  three 'filters' (truth models) so as to have at
//  least something to start with
//
//  usually the first truth model (f01X1.dat) will
//  be the fully functional case, and f02X1.dat and
//  f03X1.dat will hold the first single and double
//  failure truth models respectively
//
fsave 'f03X1.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
fsave 'f02X1.dat' ...
```

```
fbk aa bb phix bd cqd h gkf r ak akinv detak
fsave 'f01X1.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 1 failure filter, #05
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,9)=cz14;
b(:,1)=cz14;
bb=bbcon;
bb(:,1)=cz8;
exec('mtx.mxx')
fsave 'F05X1.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 2 failure filter, #06
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,10)=cz14;
b(:,2)=cz14;
bb=bbcon;
bb(:,2)=cz8;
exec('mtx.mxx')
fsave 'F06X1.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 3 failure filter, #07
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,11)=cz14;
b(:,3)=cz14;
bb=bbcon;
bb(:,3)=cz8;
exec('mtx.mxx')
fsave 'F07X1.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 4 failure filter, #08
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,12)=cz14;
b(:,4)=cz14;
bb=bbcon;
bb(:,4)=cz8;
exec('mtx.mxx')
fsave 'F08X1.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 5 failure filter, #09
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
```

```
//f(:,13)=cz14;
b(:,5)=cz14;
bb=bbcon;
bb(:,5)=cz8;
exec('mtx.mxx')
fsave 'F09X1.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 1 failure filter, #10
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(1,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F10X1.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 2 failure filter, #11
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(2,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F11X1.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 3 failure filter, #12
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(3,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F12X1.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 4 failure filter, #13
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(4,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F13X1.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 5 failure filter, #14
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(5,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F14X1.dat' ...
```

```
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 6 failure filter, #15
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(6,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F15X1.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 7 failure filter, #16
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(7,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F16X1.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//
//  actuator 6 failure filter, #17
//
//  this has been removed, but can be uncommented out
//  originally, the failed act 6 filter was #10,
//  but is now appended on to the end (as filter #17)
//  if used.
//
//fbk=fbk+.01;
//f=fcon;
//b=bcon;
//h=hcon;
////f(:,14)=cz14;
//b(:,6)=cz14;
//bb=bbcon;
//bb(:,6)=cz8;
//exec('mtx.mxx')
//fsave 'F10X1.dat' ...
//fbk aa bb phix bd cqd h gkf r ak akinv detak
//
return
//
//
//↑
```

```
//
//          SETUPX2.MXX MATRIXX EXECUTABLE MACRO
//
//                Author:  Capt Gregory L. Stratton
//          Date created:  20 August 1991
//          Date revised:  26 September 1991
//
//  This macro creates and saves to files all the required
//  matrices for a single bank, as used in MMAESIM.  This
//  is normally called from the macro FILECREATE.MXX, however,
//  it can be used by itself as long as the below listed
//  variables currently exist in memory in matrixx.
//
//  Below are listed the required input matrices and variables:
//
//     aa,      The 8x8 plant matrix
// bbcon,      The 8x6 plant B matrix
//      g,      The white Gaussian noise multiplier matrix (as in G*w(t))
//     gd,      An identity matrix of size 14x14
//      q,      The white Gaussian noise covariance matrix
//  fcon,      The 14x14 plant matrix augmented with 1st order actuators
//  bcon,      The 14x6 B matrix of the augmented system
//  hcon,      The 7x14 H matrix of the measurement equation
//  delt,      The sample time (here 1/64 Hz)
//      n,      number of states of the augmented system (here 14)
//  bank,      number of bank of which filters are being created
//  cz14,      column vector of 14 zeros
//   cz8,      column vector of 8 zeros
//  rz14,      row vector of 14 zeros
//
//  Note:  any of the variables above with 'con' in its name may
//         contain rows or columns of zeros, simulating an already
//         detected first failure.
//
//  This macro creates 14 files of the form:  FxxX2.DAT
//  where xx is the filter number (04 thru 17)
//  This macro calls the macro "mtx.mxx" which generates the required
//  matrices for each filter.
//
//
//  S T A R T   M A C R O
//
//
//  fully functional filter, #04
//
fbk=.04+bank/10000;
f=faug;
b=baug;
bb=borig;
h=horig;
exec('mtx.mxx')
fsave 'F04X2.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  also save the fully functional case to the first
//  three 'filters' (truth models) so as to have at
//  least something to start with
//
//  usually the first truth model (f01X2.dat) will
//  be the fully functional case, and f02X2.dat and
//  f03X2.dat will hold the first single and double
//  failure truth models respectively
//
fsave 'f03X2.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
fsave 'f02X2.dat' ...
```

```
fbk aa bb phix bd cqd h gkf r ak akinv detak
fsave 'f01X2.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 1 failure filter, #05
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,9)=cz14;
b(:,1)=cz14;
bb=bbcon;
bb(:,1)=cz8;
exec('mtx.mxx')
fsave 'F05X2.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 2 failure filter, #06
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,10)=cz14;
b(:,2)=cz14;
bb=bbcon;
bb(:,2)=cz8;
exec('mtx.mxx')
fsave 'F06X2.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 3 failure filter, #07
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,11)=cz14;
b(:,3)=cz14;
bb=bbcon;
bb(:,3)=cz8;
exec('mtx.mxx')
fsave 'F07X2.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 4 failure filter, #08
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,12)=cz14;
b(:,4)=cz14;
bb=bbcon;
bb(:,4)=cz8;
exec('mtx.mxx')
fsave 'F08X2.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 5 failure filter, #09
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
```

```
//f(:,13)=cz14;
b(:,5)=cz14;
bb=bbcon;
bb(:,5)=cz8;
exec('mtx.mxx')
fsave 'F09X2.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 1 failure filter, #10
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(1,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F10X2.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 2 failure filter, #11
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(2,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F11X2.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 3 failure filter, #12
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(3,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F12X2.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 4 failure filter, #13
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(4,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F13X2.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 5 failure filter, #14
//
fbk=fbk+.01;
f=fcon:
b=bcon;
h=hcon;
h(5,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F14X2.dat' ...
```

```
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 6 failure filter, #15
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(6,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F15X2.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 7 failure filter, #16
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(7,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F16X2.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//
//  actuator 6 failure filter, #17
//
//  this has been removed, but can be uncommented out
//  originally, the failed act 6 filter was #10,
//  but is now appended on to the end (as filter #17)
//  if used.
//
//fbk=fbk+.01;
//f=fcon;
//b=bcon;
//h=hcon;
////f(:,14)=cz14;
//b(:,6)=cz14;
//bb=bbcon;
//bb(:,6)=cz8;
//exec('mtx.mxx')
//fsave 'F10X2.dat' ...
//fbk aa bb phix bd cqd h gkf r ak akinv detak
//
return
//
//
//
```

```
//
//        SETUPX3.MXX MATRIXX EXECUTABLE MACRO
//
//              Author:  Capt Gregory L. Stratton
//        Date created:  20 August 1991
//        Date revised:  26 September 1991
//
//  This macro creates and saves to files all the required
//  matrices for a single bank, as used in MMAESIM.  This
//  is normally called from the macro FILECREATE.MXX, however,
//  it can be used by itself as long as the below listed
//  variables currently exist in memory in matrixx.
//
//  Below are listed the required input matrices and variables:
//
//     aa,     The 8x8 plant matrix
// bbcon,     The 8x6 plant B matrix
//      g,     The white Gaussian noise multiplier matrix (as in G*w(t))
//     gd,     An identity matrix of size 14x14
//      q,     The white Gaussian noise covariance matrix
//  fcon,     The 14x14 plant matrix augmented with 1st order actuators
//  bcon,     The 14x6 B matrix of the augmented system
//  hcon,     The 7x14 H matrix of the measurement equation
//  delt,     The sample time (here 1/64 Hz)
//      n,     number of states of the augmented system (here 14)
//  bank,     number of bank of which filters are being created
//  cz14,     column vector of 14 zeros
//   cz8,     column vector of 8 zeros
//  rz14,     row vector of 14 zeros
//
//  Note:  any of the variables above with 'con' in its name may
//         contain rows or columns of zeros, simulating an already
//         detected first failure.
//
//  This macro creates 14 files of the form:  FxxX3.DAT
//  where xx is the filter number (04 thru 17)
//  This macro calls the macro "mtx.mxx" which generates the required
//  matrices for each filter.
//
//
//  S T A R T   M A C R O
//
//
//  fully functional filter, #04
//
fbk=.04+bank/10000;
f=faug;
b=baug;
bb=borig;
h=horig;
exec('mtx.mxx')
fsave 'F04X3.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  also save the fully functional case to the first
//  three 'filters' (truth models) so as to have at
//  least something to start with
//
//  usually the first truth model (f01X3.dat) will
//  be the fully functional case, and f02X3.dat and
//  f03X3.dat will hold the first single and double
//  failure truth models respectively
//
fsave 'f03X3.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
fsave 'f02X3.dat' ...
```

```
fbk aa bb phix bd cqd h gkf r ak akinv detak
fsave 'f01X3.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 1 failure filter, #05
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,9)=cz14;
b(:,1)=cz14;
bb=bbcon;
bb(:,1)=cz8;
exec('mtx.mxx')
fsave 'F05X3.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 2 failure filter, #06
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,10)=cz14;
b(:,2)=cz14;
bb=bbcon;
bb(:,2)=cz8;
exec('mtx.mxx')
fsave 'F06X3.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 3 failure filter, #07
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,11)=cz14;
b(:,3)=cz14;
bb=bbcon;
bb(:,3)=cz8;
exec('mtx.mxx')
fsave 'F07X3.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 4 failure filter, #08
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
//f(:,12)=cz14;
b(:,4)=cz14;
bb=bbcon;
bb(:,4)=cz8;
exec('mtx.mxx')
fsave 'F08X3.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  actuator 5 failure filter, #09
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
```

```
//f(:,13)=cz14;
b(:,5)=cz14;
bb=bbcon;
bb(:,5)=cz8;
exec('mtx.mxx')
fsave 'F09X3.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 1 failure filter, #10
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(1,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F10X3.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 2 failure filter, #11
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(2,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F11X3.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 3 failure filter, #12
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(3,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F12X3.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 4 failure filter, #13
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(4,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F13X3.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 5 failure filter, #14
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(5,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F14X3.dat' ...
```

```
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 6 failure filter, #15
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(6,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F15X3.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//  sensor 7 failure filter, #16
//
fbk=fbk+.01;
f=fcon;
b=bcon;
h=hcon;
h(7,:)=rz14;
bb=bbcon;
exec('mtx.mxx')
fsave 'F16X3.dat' ...
fbk aa bb phix bd cqd h gkf r ak akinv detak
//
//
//  actuator 6 failure filter, #17
//
//  this has been removed, but can be uncommented out
//  originally, the failed act 6 filter was #10,
//  but is now appended on to the end (as filter #17)
//  if used.
//
//fbk=fbk+.01;
//f=fcon;
//b=bcon;
//h=hcon;
////f(:,14)=cz14;
//b(:,6)=cz14;
//bb=bbcon;
//bb(:,6)=cz8;
//exec('mtx.mxx')
//fsave 'F10X3.dat' ...
//fbk aa bb phix bd cqd h gkf r ak akinv detak
//
return
//
//
//
```

APPENDIX E: DATA FILES

This appendix contains all of the data files necessary for the execution of the MMAESIM code. The data files necessary for the proper execution of the code are provided within this section. The filter single-failure data files are not presented in this section, with the exception of the no-failure filter. Since there are 16 filters and 13 banks (208 data filters), presenting these filters within this appendix is impractical. The no-failure filter data is presented to represent the format of the filter data files.

```
C***********************************************************
C                                                          *
C     LOGICAL CONTROL FILE - [REALS.DAT] is called from    *
C     SUBROUNTINE GETDAT.  REALS.DAT functions as an       *
C     input data set which provides the necessary real     *
C     variables and/or initial settings for MMAESIM        *
C     to begin.  By placing all of the starting real       *
C     variables within a single logical file, it becomes   *
C     easy to adjust the simulation for any possible       *
C     scenario without recompiling the code.  A single     *
C     locale for all of the starting real variables        *
C     provides the user with a quick reference of the      *
C     scenario under investigation.                        *
C                                                          *
C     CREATION DATE:   8 July 1991                         *
C     REVISION DATE:   8 July 1991                         *
C     Owner:  USAF/ASD/WL/AFIT                             *
C                                                          *
C***********************************************************
C
C
C
C
C  ------------------------------------------------------  *
C               MMAESIM REAL CONTROL VARIABLES             *
C  ------------------------------------------------------  *
C
C
C  VAR -> TSAMP             :  Sampling time used in MMAESIM.
C                              Currently this is set to run at
C                              a 64 Hz rate, which is
C                              TSAMP = 1/(64 Hz).
C
    TSAMP = 0.015625
C
C  ........................................................
C
C  VAR -> PRBMIN            :  Minimum allowed probability for
C                              any one filter.
C
    PRBMIN = 0.001
C
C  ........................................................
C
C  VAR -> PRBFLTRT0         :  Probability given to the assumed
C                              "correct" filter at time zero.
C                              Each of the other remaining filters
C                              are assigned a probability of
C                              (1-PRBFLTRT0)/(#filters-1)
C
    PRBFLTRT0 = 0.75
C
C  ........................................................
C
C  VAR -> DSIM              :  Simulation running time in seconds.
C                              Current maximum DSIM is 15.625 seconds.
C
    DSIM = 8.0
C
C  ........................................................
C
C  VAR -> WGNFAC            :  White Gaussian noise factor.
C
    WGNFAC = 0.05
C
C  ........................................................
```

```
C
C  VAR -> ZBIASAMNT        :  The array of bias amounts that may
C                             corrupt the measurements.  Currently,
C                             there are 7 measurements used.
C                             Normally, if a bias error is being
C                             simulated only one of the measurements
C                             would be corrupted with a bias, and
C                             the others would be zero.  Below are
C                             the "suggested" bias amounts for each
C                             of the sensor measurements.
C
C  VELOCITY      <-->  ZBIASAMNT(1) =
C  ALPHA         <-->  ZBIASAMNT(2) =
C  PITCH RATE    <-->  ZBIASAMNT(3) =
C  LOAD FACTOR   <-->  ZBIASAMNT(4) =
C  ROLL RATE     <-->  ZBIASAMNT(5) =
C  YAW RATE      <-->  ZBIASAMNT(6) =
C  LAT. ACCEL.   <-->  ZBIASAMNT(7) =
C
    ZBIASAMNT(1) = 0.0
    ZBIASAMNT(2) = 0.0
    ZBIASAMNT(3) = 0.0
    ZBIASAMNT(4) = 0.0
    ZBIASAMNT(5) = 0.0
    ZBIASAMNT(6) = 0.0
    ZBIASAMNT(7) = 0.0
C
C  ..............  ......................................
C
C  VAR -> TIMELAG1         :  Time in which the first failure
C                             occurs.  This must be greater than
C                             or equal to zero and less than
C                             TIMELAG2 (if 2nd failure is being
C                             modeled).
C
    TIMELAG1 = 3.0
C
C  ......................................................
C
C  VAR -> TIMELAG2         :  Time in which the second failure
C                             occurs.  This must be greater than
C                             or equal to TIMELAG1.
C
    TIMELAG2 = 3.1
C
C  ......................................................
C
C  VAR -> DSEED            :  Seed provided to random number
C                             generator.
C
C   DSEED = 754.5463
C
C  ......................................................
C
C
C  -------------------------------------------------- *
C                TUNING VARIABLES USED IN THE Q MATRIX      *
C  -------------------------------------------------- *
C
C                                                      --
C  TSIG11      |TSIG11   0      0      0    ...   0    |
C  TSIG22      |       TSIG22   0      0    ...   0    |
C  TSIG33      |              TSIG33   0    ...   0    |
C  TSIG44      |                     TSIG44 ...   0    |
C  TSIG55      |                            ...   0    |
C  TSIG66      |      (  SYM  )             ...   0    |
```

```
C  TSIG77       |                                        |
C  TSIG88       |                              TSIG88    |
C              --                                       --
C
   TSIG11 = 0.0
   TSIG22 = 0.0
   TSIG33 = 0.0
   TSIG44 = 0.0
   TSIG55 = 0.0
   TSIG66 = 0.0
   TSIG77 = 0.0
   TSIG88 = 0.0
C
C
C  ------------------------------------------------------   *
C            END OF MMAESIM REAL CONTROL VARIABLES           *
C  ------------------------------------------------------   *
C
C  .........................................................
C
C  VAR -> M             :  Number of sensors.  Currently 7.
C
     M = 7.0
C
C
```

```
C*********************************************************
C                                                        *
C     LOGICAL CONTROL FILE -  [FLAGS.DAT] is called      *
C     from SUBROUTINE GETDAT.  FLAGS.DAT functions as    *
C     an input data set which controls all the logical   *
C     switching within MMAESIM.  By placing all of the   *
C     logical files within a single file, it becomes     *
C     easy to adjust the simulation for any possible     *
C     scenario without recompiling the code.  A single   *
C     locale for all of the logical flags provides the   *
C     user with a quick reference of the scenario under  *
C     investigation.                                     *
C                                                        *
C     CREATION DATE:  29 June 1991                       *
C     REVISION DATE:   1 July 1991                       *
C     Owner:  USAF/ASD/AFIT                              *
C                                                        *
C*********************************************************
C
C
C
C
C*********************************************************
C*********************************************************
C**************                           ***************
C*************          L E V E L   0     ***************
C**************                           ***************
C*********************************************************
C*********************************************************
C
C
C
C   --------------------------------------------------   *
C              MMAESIM MASTER CONTROL FLAGS              *
C   --------------------------------------------------   *
C
C
C  FLAG ->IDID      : (1) - Var A is used, (2) - Var B is used
C                   : Description - this flag is used within
C                     MMAE to switch between loop a and loop b
C
      IDID = 13
C
C
C
C
C
C  FLAG ->MODELN     : (1) - Var A is used, (2) - Var B is used
C                   : Description - this flag is used within
C                     MMAE to switch between loop a and loop b
C
      MODELN = 1
C
C
C
C  FLAG ->MODELN1     : (1) - Var A is used, (2) - Var B is used
C                   : Description - this flag is used within
C                     MMAE to switch between loop a and loop b
C
      MODELN1 = 1
C
C
C  FLAG ->MODELN2     : (1) - Var A is used, (2) - Var B is used
C                   : Description - this flag is used within
C                     MMAE to switch between loop a and loop b
C
```

```
      MODELN2 = 2
C
C
C  FLAG ->MODELN3      : (1) - Var A is used, (2) - Var B is used
C                     : Description - this flag is used within
C                       MMAE to switch between loop a and loop b
C
      MODELN3 = 3
C
C
C
C  FLAG ->ISTART       : (1) - Var A is used, (2) - Var B is used
C                     : Description - this flag is used within
C                       MMAE to switch between loop a and loop b
C
      ISTART = 0
C
C
C
C  FLAG ->WFLAG        : (1) - Var A is used, (2) - Var B is used
C                     : Description - this flag is used within
C                       MMAE to switch between loop a and loop b
C
      WFLAG = 1
C
C
C
C  FLAG ->NUMFAILS      : (1) - Var A is used, (2) - Var B is used
C                     : Description - this flag is used within
C                       MMAE to switch between loop a and loop b
C
      NUMFAILS = 2
C
C
C
C  FLAG ->NUMBANKS      : (1) - Var A is used, (2) - Var B is used
C                     : Description - this flag is used within
C                       MMAE to switch between loop a and loop b
C
      NUMBANKS = 13
C
C
C
C  FLAG ->BANK        : (1) - Var A is used, (2) - Var B is used
C                     : Description - this flag is used within
C                       MMAE to switch between loop a and loop b
C
      BANK = 1
C
C
C
C  FLAG ->HIERARCHY      : (1) - Var A is used, (2) - Var B is used
C                     : Description - this flag is used within
C                       MMAE to switch between loop a and loop b
C
      HIERARCHY = 0
C
C
C
C
C  FLAG ->XITER        : (1) - Var A is used, (2) - Var B is used
C                     : Description - this flag is used within
C                       MMAE to switch between loop a and loop b
C
      XITER = 1
C
```

```
C
C  FLAG ->SENSORBIAS    : (1) - Var A is used, (2) - Var B is used
C                    : Description - this flag is used within
C                      MMAE to switch between loop a and loop b
C
      SENSORBIAS = 0
C
C
C
C  FLAG ->FLTRT0     : (1) - Var A is used, (2) - Var B is used
C                    : Description - this flag is used within
C                      MMAE to switch between loop a and loop b
C
      FLTRT0 = 4
C
C
C  FLAG ->NFLTR      : (1) - Var A is used, (2) - Var B is used
C                    : Description - this flag is used within
C                      MMAE to switch between loop a and loop b
C
      NFLTR(1) = 16
      NFLTR(2) = 16
      NFLTR(3) = 16
      NFLTR(4) = 16
      NFLTR(5) = 16
      NFLTR(6) = 16
      NFLTR(7) = 16
      NFLTR(8) = 16
      NFLTR(9) = 16
      NFLTR(10) = 16
      NFLTR(11) = 16
      NFLTR(12) = 16
      NFLTR(13) = 16
      NFLTR(14) = 0
      NFLTR(15) = 0
C
C
C
C
C
C************************************************************
C************************************************************
C**************                              ***************
C**************        L E V E L   1          ***************
C**************                              ***************
C************************************************************
C************************************************************
C
C
C
C  -----------------------------------------------------   *
C        SUBROUTINE  [GETDAT]  CONTROL FLAGS               *
C  -----------------------------------------------------   *
C
C
C
C
C
C  -----------------------------------------------------   *
C        SUBROUTINE  [PULLOUT]  CONTROL FLAGS              *
C  -----------------------------------------------------   *
C
C
C
C
C  -----------------------------------------------------   *
```

```
C          SUBROUTINE  [MATML]  CONTROL FLAGS                 *
C   -----------------------------------------------------    *
C
C
C
C
C
C   -----------------------------------------------------    *
C          SUBROUTINE  [GGNML]  CONTROL FLAGS                 *
C   -----------------------------------------------------    *
C
C
C
C
C
C   -----------------------------------------------------    *
C          SUBROUTINE  [KFILT]  CONTROL FLAGS                 *
C   -----------------------------------------------------    *
C
C
C
C   FLAG ->BANKFLAG    : (1) - Var A is used, (2) - Var B is used
C                   : Description - this flag is used within
C                     MMAE to switch between loop a and loop b
C
      BANKFLAG = 0
C
C
C
C   FLAG ->INITV2     : (1) - Var A is used, (2) - Var B is used
C                   : Description - this flag is used within
C                     MMAE to switch between loop a and loop b
C
      INITV2 = 0
C
C
C   FLAG ->INITV    : (1) - Var A is used, (2) - Var B is used
C                   : Description - this flag is used within
C                     MMAE to switch between loop a and loop b
C
      INITV = 1
C
C
C
C
C   -----------------------------------------------------    *
C          SUBROUTINE  [DEABM]  CONTROL FLAGS                 *
C   -----------------------------------------------------    *
C
C
C   -----------------------------------------------------    *
C          SUBROUTINE  [PLTLSR]  CONTROL FLAGS                *
C   -----------------------------------------------------    *
C
C
C
C
C
C   -----------------------------------------------------    *
C        SUBROUTINE  [TIME AND DATE] CONTROL FLAGS            *
C   -----------------------------------------------------    *
C
C
C
C
C
```

```
C
C*************************************************************
C*************************************************************
C**************                              ***************
C**************        L E V E L   2         ***************
C**************                              ***************
C*************************************************************
C*************************************************************
C
C
C
C
C  -----------------------------------------------------   *
C           SUBROUTINE  [EOM]  CONTROL FLAGS               *
C  -----------------------------------------------------   *
C
C
C
C
C  FLAG ->IACTORDR     : (1) - First order actuator model, (2) - Second
C                        order actuator model, (4) - Fourth order actuator
C                        model
C                      : Description - this flag is used within EOM to
C                        select the actuator model.
C
      IACTORDR = 1
C
C
C
C
C
C  -----------------------------------------------------   *
C         SUBROUTINE  (UPDATE)  CONTROL FLAGS              *
C  -----------------------------------------------------   *
C
C
C  FLAG ->BETAFLG     : (1) - Var A is used, (2) - Var B is used
C                      : Description - this flag is used within
C                        MMAE to switch between loop a and loop b
C
      BETAFLG = 0
C
C
C
C
C  -----------------------------------------------------   *
C         SUBROUTINE  [CNTRL]   CONTROL FLAGS              *
C  -----------------------------------------------------   *
C
C
C  FLAG ->ITRIMZ     : (0) - Read trim file, (1) - do not read trim file
C                   : Description - this flag is used within CNTRL to
C                     read the trim file for controls initialization
C
      ITRIMZ = 0
C
C
C
C
C  -----------------------------------------------------   *
C         SUBROUTINE  [ADPCON]  CONTROL FLAGS              *
C  -----------------------------------------------------   *
C
C
C  FLAG ->ISLCT     : (1) - Var A is used, (2) - Var B is used
C                   : Description - this flag is used within
```

```
C                        MMAE to switch between loop a and loop b
C
      ISLCT = 3
C
C
C
C  FLAG ->DSEED      : (1) - Var A is used, (2) - Var B is used
C                    : Description - this flag is used within
C                      MMAE to switch between loop a and loop b
C
      DSEED = -6
C
C
C
C
C
C***************************************************************
C***************************************************************
C**************                               ****************
C*************            L E V E L   3        **************
C**************                               ****************
C***************************************************************
C***************************************************************
C
C
C
C
C  ----------------------------------------------------------  *
C         SUBROUTINE  [DSORT]  CONTROL FLAGS                   *
C  ----  ----------------------------------------------------  *
C
C
C
C  ----------------------------------------------------------  *
C         SUBROUTINE  [ACALC]  CONTROL FLAGS                   *
C  ----------------------------------------------------------  *
C
      IACTFAIL = 0
C
      IACTFL2 = 5
C
C
C
C
```

```
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C  DECLARATION OF VARIABLES AND COMMON BLOCKS FOR MMAESIM
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

      REAL  Z(7),H(7,14),HT(7,29)
      REAL  CQDCNT(8,8),R(7,7),SMPLS
      DOUBLE PRECISION  PI
      INTEGER XITER,INITV,IACTORDR,DSEED,IACTFAIL,IACTFL2
      CHARACTER CDATE*9,CTIME*8

      REAL TSIG11,TSIG22,TSIG33,TSIG44,TSIG55,TSIG66,TSIG77,TSIG88
      REAL A(8,8),B(8,6),C(29,29)

      REAL AUNEW(6),PRBNEW(20,15),PRBMIN,PRBFLTRT0,TSAMP
      REAL XHPSUM(14),DSIM,WGNFAC,TIMELAG1,TIMELAG2

      INTEGER NFLTR(15),ISLCT
      INTEGER MODELN,Modeln1,Modeln2,Modeln3
      INTEGER ISTART,ISTOP,FLTRT0,BETAFLG,WFLAG
      DOUBLE PRECISION M

      DOUBLE PRECISION OUT(513,29),DEFLEC(513,6),INPUTS(513,6)
      DOUBLE PRECISION STATES(513,8),PROBS(513,17),TVEC(513,1)
      DOUBLE PRECISION DUMMYJ,ACCEL(513,2),PRBBNK2(513,17)
      DOUBLE PRECISION RSID(192,91),TSHORT(192,1),BDUSG(192,91)
      DOUBLE PRECISION RSIDTWO(192,91),BDUSGTWO(192,91)

C --- Elemental Filter Data Arrays

      REAL ZA(8,8,20,15),ZB(8,6,20,15)
      REAL ZPHIX(14,14,20,15),ZBD(14,6,20,15),ZCQDCN(8,8,20,15)
      REAL ZH(7,14,20,15),ZGKF(14,7,20,15),ZR(7,7,20,15)
      REAL ZAK(7,7,20,15),ZAKINV(7,7,20,15),ZDETAK(20,15)

C --- Residual and Log Likelihood Declarations

      INTEGER itime,ijk
      DOUBLE PRECISION rakr(513,7,10,20)
C     DOUBLE PRECISION LK(513,7,20)
      DOUBLE PRECISION rssave(513,7,13)
      DOUBLE PRECISION buzsave(513,7,13)
c     REAL blzsave(513,7,13)


C --- Sequential and Multiple Failure Variables

      INTEGER numfails

C --- Sensor Bias Variables

      INTEGER Sensorbias
      REAL Zbiasamnt(7)

C --- Hierarchical Modeling Variables

      CHARACTER*5 DFILE(20,15)
      CHARACTER*5 BANKNAME(15)
      INTEGER Numbanks,Bank,Bankflag,initv2

C --- Mean and Standard Deviation Variables

      INTEGER ki,kj,kk
      REAL Temp1,Temp2
      DOUBLE PRECISION Meanprob(3,16),Stddev(3,16)

C --- Code Checking algorithms
```

```
c      Real Xdifstat(     ),Xdifcon(     )
c      Real Zdifstat(     ),Zdifcon(     )
c      Real h


c . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
c     COMMON BLOCK DEFINITIONS
c . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

      COMMON/LOGFLAGS/IDID,MODELN,MODELN1,MODELN2,MODELN3,ISTART,
     &WFLAG,NUMFAILS,NUMBANKS,BANK,HIERARCHY,XITER,SENSORBIAS,
     &FLTRT0,NFLTR,BANKFLAG,INITV2,INITV,IACTORDR,BETAFLG,ITRIMZ,
     &ISLCT,DSEED,IACTFAIL,IACTFL2

      COMMON/RAWDAT/ZA,ZB,ZPHIX,ZBD,ZCQDCN,ZH,ZGKF,ZR,
     &ZAK,ZAKINV,ZDETAK,CQDCNT,R

      COMMON/restst/rakr,ijk,itime,rssave,buzsave

      COMMON/CHRDAT/DFILE,BANKNAME

      COMMON/CONTROLSZ/AUNEW,PRBNEW,XHPSUM,Z,H,HT

      COMMON/MATRIX/A,B,C

      COMMON/PLTINF/OUT,DEFLEC,INPUTS,STATES,PROBS,PRBBNK2

      COMMON/STATS/MEANPROB,STDDEV

      COMMON/REALDAT/TSAMP,PRBMIN,PRBFLTRT0,DSIM,WGNFAC,ZBIASAMNT,
     &TIMELAG1,TIMELAG2,TSIG11,TSIG22,TSIG33,TSIG44,TSIG55,TSIG66,
     &TSIG77,TSIG88,M
```

414.7300	1.2665000E-03	972.5400	2116.200	0.4000000
3.0071932E-04	20000.00	0.0000000E+00	0.0000000E+00	3.0071932E-04
8.300000	-2.587025	3.0071911E-04	3.0071911E-04	3.0071932E-04
8.300000	1.5625000E-02	0.0000000E+00	0.0000000E+00	0.0000000E+00
0.0000000E+00	10.06088	22.02900	0.0000000E+00	10.06088
10.06088	10.06088	10.06088	10.00000	10.06088
0.4200000	4.225569	0.9999998	0.9999998	1.000000
1.000000	2.3841858E-07	0.0000000E+00	0.0000000E+00	0.0000000E+00
0.0000000E+00	0.0000000E+00	6.3923690E-03	10.06088	0.6499329
-1.847997	0.0000000E+00	0.6499329	0.0000000E+00	-4.9999952E-03
-4.9999952E-03	-4.9999924E-03	8.916890	-4.9999868E-03	-4.9999868E-03
-4.9999924E-03	-4.9999924E-03	0.0000000E+00	0.0000000E+00	0.0000000E+00
-5.3007058E-03	0.0000000E+00	1.260000	1.557974	0.0000000E+00
1.557974	1.557974	-3.3401489E-02	1.557974	-3.3401489E-02
1.557974	1.557974	1.557974	1.557974	1.557974
-0.4403729	-0.4403729	-6.6788890E-03	-3.3394445E-02	0.0000000E+00
-2.660917	-2.660917	-3.3394445E-02	-2.667595	1.557974
1.500000	0.0000000E+00	0.0000000E+00	0.1165518	10.06087
10.06087	10.06088	10.06088	14.40508	14.34643
-10.72676	14.40508	-2.000000	14.40508	0.0000000E+00
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00
0.0000000E+00	-3.3992767E-02	14.40509	14.40509	5090.137
5090.137	14.40509	0.0000000E+00	0.0000000E+00	1.500000
1.500000	1.500000	1.500000	1.500000	1.500000
0.0000000E+00	0.0000000E+00	1.500000	1.500000	21.00000
-23.00000	0.0000000E+00	0.0000000E+00	1.500000	1.500000
1.500000	1.500000	0.0000000E+00	1.500000	1.500000
0.0000000E+00	-3.5762787E-07	-3.5762787E-07		

```
C*************************************************************
C                                                            *
C     FILTER NAME FILE -  [FLTRNAME.DAT] is called           *
C     from SUBROUTINE GETDAT.  FLTRNAME.DAT functions as     *
C     an input data set which controls all the data file     *
C     reads for the individual Kalman filters within each    *
C     of the possible 15 banks of filters.  This input       *
C     file contains the filter names corresponding to a      *
C     a filter data file.  Subroutine REDNAM.FOR reads       *
C     this file and loads a common block CHRDAT for use      *
C     within MMAESIM.                                        *
C                                                            *
C                                                            *
C                                                            *
C     CREATION DATE:  10 July 1991                           *
C     REVISION DATE:  11 July 1991                           *
C     Owner:  USAF/ASD/AFIT                                  *
C                                                            *
C*************************************************************
C
C
C
C
C   ------------------------------------------------------   *
C                    FILTER BANK NUMBER # 1                  *
C   ------------------------------------------------------   *
C
C
      FILTER #01 = F01B1
      FILTER #02 = F02B1
      FILTER #03 = F03B1
      FILTER #04 = F04B1
      FILTER #05 = F05B1
      FILTER #06 = F06B1
      FILTER #07 = F07B1
      FILTER #08 = F08B1
      FILTER #09 = F09B1
      FILTER #10 = F10B1
      FILTER #11 = F11B1
      FILTER #12 = F12B1
      FILTER #13 = F13B1
      FILTER #14 = F14B1
      FILTER #15 = F15B1
      FILTER #16 = F16B1
      FILTER #17 = F17B1
      FILTER #18 = F18B1
      FILTER #19 = F19B1
      FILTER #20 = F20B1
C
C
C   ------------------------------------------------------   *
C                    FILTER BANK NUMBER # 2                  *
C   ------------------------------------------------------   *
C
      FILTER #01 = F01B2
      FILTER #02 = F02B2
      FILTER #03 = F03B2
      FILTER #04 = F04B2
      FILTER #05 = F05B2
      FILTER #06 = F06B2
      FILTER #07 = F07B2
      FILTER #08 = F08B2
      FILTER #09 = F09B2
      FILTER #10 = F10B2
      FILTER #11 = F11B2
      FILTER #12 = F12B2
```

```
      FILTER #13 = F13B2
      FILTER #14 = F14B2
      FILTER #15 = F15B2
      FILTER #16 = F16B2
      FILTER #17 = F17B2
      FILTER #18 = F18B2
      FILTER #19 = F19B2
      FILTER #20 = F20B2
C
C  ---------------------------------------------------------------  *
C                       FILTER BANK NUMBER # 3                       *
C  ---------------------------------------------------------------  *
C
      FILTER #01 = F01B3
      FILTER #02 = F02B3
      FILTER #03 = F03B3
      FILTER #04 = F04B3
      FILTER #05 = F05B3
      FILTER #06 = F06B3
      FILTER #07 = F07B3
      FILTER #08 = F08B3
      FILTER #09 = F09B3
      FILTER #10 = F10B3
      FILTER #11 = F11B3
      FILTER #12 = F12B3
      FILTER #13 = F13B3
      FILTER #14 = F14B3
      FILTER #15 = F15B3
      FILTER #16 = F16B3
      FILTER #17 = F17B3
      FILTER #18 = F18B3
      FILTER #19 = F19B3
      FILTER #20 = F20B3
C
C  ---------------------------------------------------------------  *
C                       FILTER BANK NUMBER # 4                       *
C  ---------------------------------------------------------------  *
C
      FILTER #01 = F01B4
      FILTER #02 = F02B4
      FILTER #03 = F03B4
      FILTER #04 = F04B4
      FILTER #05 = F05B4
      FILTER #06 = F06B4
      FILTER #07 = F07B4
      FILTER #08 = F08B4
      FILTER #09 = F09B4
      FILTER #10 = F10B4
      FILTER #11 = F11B4
      FILTER #12 = F12B4
      FILTER #13 = F13B4
      FILTER #14 = F14B4
      FILTER #15 = F15B4
      FILTER #16 = F16B4
      FILTER #17 = F17B4
      FILTER #18 = F18B4
      FILTER #19 = F19B4
      FILTER #20 = F20B4
C
C  ---------------------------------------------------------------  *
C                       FILTER BANK NUMBER # 5                       *
C  ---------------------------------------------------------------  *
C
      FILTER #01 = F01B5
      FILTER #02 = F02B5
      FILTER #03 = F03B5
```

```
      FILTER #04 = F04B5
      FILTER #05 = F05B5
      FILTER #06 = F06B5
      FILTER #07 = F07B5
      FILTER #08 = F08B5
      FILTER #09 = F09B5
      FILTER #10 = F10B5
      FILTER #11 = F11B5
      FILTER #12 = F12B5
      FILTER #13 = F13B5
      FILTER #14 = F14B5
      FILTER #15 = F15B5
      FILTER #16 = F16B5
      FILTER #17 = F17B5
      FILTER #18 = F18B5
      FILTER #19 = F19B5
      FILTER #20 = F20B5
C
C  ---------------------------------------------------------------   *
C                      FILTER BANK NUMBER # 6                        *
C  ---------------------------------------------------------------   *
C
      FILTER #01 = F01B6
      FILTER #02 = F02B6
      FILTER #03 = F03B6
      FILTER #04 = F04B6
      FILTER #05 = F05B6
      FILTER #06 = F06B6
      FILTER #07 = F07B6
      FILTER #08 = F08B6
      FILTER #09 = F09B6
      FILTER #10 = F10B6
      FILTER #11 = F11B6
      FILTER #12 = F12B6
      FILTER #13 = F13B6
      FILTER #14 = F14B6
      FILTER #15 = F15B6
      FILTER #16 = F16B6
      FILTER #17 = F17B6
      FILTER #18 = F18B6
      FILTER #19 = F19B6
      FILTER #20 = F20B6
C
C  ---------------------------------------------------------------   *
C                      FILTER BANK NUMBER # 7                        *
C  ---------------------------------------------------------------   *
C
      FILTER #01 = F01B7
      FILTER #02 = F02B7
      FILTER #03 = F03B7
      FILTER #04 = F04B7
      FILTER #05 = F05B7
      FILTER #06 = F06B7
      FILTER #07 = F07B7
      FILTER #08 = F08B7
      FILTER #09 = F09B7
      FILTER #10 = F10B7
      FILTER #11 = F11B7
      FILTER #12 = F12B7
      FILTER #13 = F13B7
      FILTER #14 = F14B7
      FILTER #15 = F15B7
      FILTER #16 = F16B7
      FILTER #17 = F17B7
      FILTER #18 = F18B7
      FILTER #19 = F19B7
```

```
      FILTER #20 = F20B7
C
C     ------------------------------------------------------------   *
C                       FILTER BANK NUMBER # 8                       *
C     ------------------------------------------------------------   *
C
      FILTER #01 = F01B8
      FILTER #02 = F02B8
      FILTER #03 = F03B8
      FILTER #04 = F04B8
      FILTER #05 = F05B8
      FILTER #06 = F06B8
      FILTER #07 = F07B8
      FILTER #08 = F08B8
      FILTER #09 = F09B8
      FILTER #10 = F10B8
      FILTER #11 = F11B8
      FILTER #12 = F12B8
      FILTER #13 = F13B8
      FILTER #14 = F14B8
      FILTER #15 = F15B8
      FILTER #16 = F16B8
      FILTER #17 = F17B8
      FILTER #18 = F18B8
      FILTER #19 = F19B8
      FILTER #20 = F20B8
C
C     ------------------------------------------------------------   *
C                       FILTER BANK NUMBER # 9                       *
C     ------------------------------------------------------------   *
C
      FILTER #01 = F01B9
      FILTER #02 = F02B9
      FILTER #03 = F03B9
      FILTER #04 = F04B9
      FILTER #05 = F05B9
      FILTER #06 = F06B9
      FILTER #07 = F07B9
      FILTER #08 = F08B9
      FILTER #09 = F09B9
      FILTER #10 = F10B9
      FILTER #11 = F11B9
      FILTER #12 = F12B9
      FILTER #13 = F13B9
      FILTER #14 = F14B9
      FILTER #15 = F15B9
      FILTER #16 = F16B9
      FILTER #17 = F17B9
      FILTER #18 = F18B9
      FILTER #19 = F19B9
      FILTER #20 = F20B9
C
C     ------------------------------------------------------------   *
C                       FILTER BANK NUMBER # 10                      *
C     ------------------------------------------------------------   *
C
      FILTER #01 = F01X0
      FILTER #02 = F02X0
      FILTER #03 = F03X0
      FILTER #04 = F04X0
      FILTER #05 = F05X0
      FILTER #06 = F06X0
      FILTER #07 = F07X0
      FILTER #08 = F08X0
      FILTER #09 = F09X0
      FILTER #10 = F10X0
```

```
      FILTER #11 = F11X0
      FILTER #12 = F12X0
      FILTER #13 = F13X0
      FILTER #14 = F14X0
      FILTER #15 = F15X0
      FILTER #16 = F16X0
      FILTER #17 = F17X0
      FILTER #18 = F18X0
      FILTER #19 = F19X0
      FILTER #20 = F20X0
C
C  -------------------------------------------------------------------  *
C                     FILTER BANK NUMBER # 11                           *
C  -------------------------------------------------------------------  *
C
      FILTER #01 = F01X1
      FILTER #02 = F02X1
      FILTER #03 = F03X1
      FILTER #04 = F04X1
      FILTER #05 = F05X1
      FILTER #06 = F06X1
      FILTER #07 = F07X1
      FILTER #08 = F08X1
      FILTER #09 = F09X1
      FILTER #10 = F10X1
      FILTER #11 = F11X1
      FILTER #12 = F12X1
      FILTER #13 = F13X1
      FILTER #14 = F14X1
      FILTER #15 = F15X1
      FILTER #16 = F16X1
      FILTER #17 = F17X1
      FILTER #18 = F18X1
      FILTER #19 = F19X1
      FILTER #20 = F20X1
C
C  -------------------------------------------------------------------  *
C                     FILTER BANK NUMBER # 12                           *
C  -------------------------------------------------------------------  *
C
      FILTER #01 = F01X2
      FILTER #02 = F02X2
      FILTER #03 = F03X2
      FILTER #04 = F04X2
      FILTER #05 = F05X2
      FILTER #06 = F06X2
      FILTER #07 = F07X2
      FILTER #08 = F08X2
      FILTER #09 = F09X2
      FILTER #10 = F10X2
      FILTER #11 = F11X2
      FILTER #12 = F12X2
      FILTER #13 = F13X2
      FILTER #14 = F14X2
      FILTER #15 = F15X2
      FILTER #16 = F16X2
      FILTER #17 = F17X2
      FILTER #18 = F18X2
      FILTER #19 = F19X2
      FILTER #20 = F20X2
C
C  -------------------------------------------------------------------  *
C                     FILTER BANK NUMBER # 13                           *
C  -------------------------------------------------------------------  *
C
      FILTER #01 = F01X3
```

```
      FILTER #02 = F02X3
      FILTER #03 = F03X3
      FILTER #04 = F04X3
      FILTER #05 = F05X3
      FILTER #06 = F06X3
      FILTER #07 = F07X3
      FILTER #08 = F08X3
      FILTER #09 = F09X3
      FILTER #10 = F10X3
      FILTER #11 = F11X3
      FILTER #12 = F12X3
      FILTER #13 = F13X3
      FILTER #14 = F14X3
      FILTER #15 = F15X3
      FILTER #16 = F16X3
      FILTER #17 = F17X3
      FILTER #18 = F18X3
      FILTER #19 = F19X3
      FILTER #20 = F20X3
C
C  -------------------------------------------------------------------  *
C                        FILTER BANK NUMBER # 14                        *
C  -------------------------------------------------------------------  *
C
      FILTER #01 = F01X4
      FILTER #02 = F02X4
      FILTER #03 = F03X4
      FILTER #04 = F04X4
      FILTER #05 = F05X4
      FILTER #06 = F06X4
      FILTER #07 = F07X4
      FILTER #08 = F08X4
      FILTER #09 = F09X4
      FILTER #10 = F10X4
      FILTER #11 = F11X4
      FILTER #12 = F12X4
      FILTER #13 = F13X4
      FILTER #14 = F14X4
      FILTER #15 = F15X4
      FILTER #16 = F16X4
      FILTER #17 = F17X4
      FILTER #18 = F18X4
      FILTER #19 = F19X4
      FILTER #20 = F20X4
C
C  -------------------------------------------------------------------  *
C                        FILTER BANK NUMBER # 15                        *
C  -------------------------------------------------------------------  *
C
      FILTER #01 = F01X5
      FILTER #02 = F02X5
      FILTER #03 = F03X5
      FILTER #04 = F04X5
      FILTER #05 = F05X5
      FILTER #06 = F06X5
      FILTER #07 = F07X5
      FILTER #08 = F08X5
      FILTER #09 = F09X5
      FILTER #10 = F10X5
      FILTER #11 = F11X5
      FILTER #12 = F12X5
      FILTER #13 = F13X5
      FILTER #14 = F14X5
      FILTER #15 = F15X5
      FILTER #16 = F16X5
      FILTER #17 = F17X5
```

```
         FILTER #18 - F18X5
         FILTER #19 - F19X5
         FILTER #20 - F20X5
C
C
C
```

```
MATRIXx VERSION 700  12     25-MAR-92 17:35
FBK             1     1AA                8     8BB               8     6PHIX           14
BD             14     6CQD               8     8H                7    14GKP            14
R               7     7AK                7     7AKINV            7     7DETAK           1
FBK             1     1     0(1P3E25.17)
  4.01000000000000002E-02
AA              8     8     0(1P3E25.17)
  0.00000000000000000E+00 -3.16783408293267712E+01 -8.44853275339119136E-07
  3.00734463962726295E-05  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  9.49566542320212648E-03 -3.80362214481788641E-04 -8.14469540522111402E-05
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  1.44258245535893366E+01
 -4.40696165424014907E-01  1.83406917617685394E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  1.00000000000000000E+00 -7.25272726987999992E+01  9.98203006730200001E-01
 -5.23819999999999994E-01  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  7.63826565105887832E-02 -5.68795346153061487E-05
  9.36393312728212024E-06  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
 -1.08260019809884830E-01 -1.82480000026640607E+01  2.80440000038280796E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  1.00000000000000000E+00  1.75468452691500000E-01
 -1.52420000000000000E+00 -9.31550000000000017E-02  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  1.77420000000000001E-01 -9.82417115568100005E-01  3.01780000000000000E-01
 -2.65570000000000001E-01
BB              8     6     0(1P3E25.17)
  0.00000000000000000E+00  1.07091049868334542E+00 -3.36500119062073796E-02
 -1.82429254242000000E+00  0.00000000000000000E+00 -6.76043593630387023E-03
  4.52407636800000001E+00  5.32077446699999998E-01  0.00000000000000000E+00
  1.07091049868334542E+00 -3.36500119062073796E-02 -1.82429254242000000E+00
  0.00000000000000000E+00  6.76043593630387023E-03 -4.52407636800000001E+00
 -5.32077446699999998E-01  0.00000000000000000E+00 -5.25680152905959930E-01
 -3.24266639757173096E-02  3.15754289010000001E-01  0.00000000000000000E+00
 -2.90307183711702749E-04  6.19969203899999999E+00  7.77899347020000013E-02
  0.00000000000000000E+00 -5.25680152905959930E-01 -3.24266639757173096E-02
  3.15754289010000001E-01  0.00000000000000000E+00  2.90307183711702749E-04
 -6.19969203899999999E+00 -7.77899347020000013E-02  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  1.66997001330417599E-02  2.84100651341999993E+00
 -1.16261199611999999E+00  0.00000000000000000E+00  1.44580000000000000E+00
  5.39999999999999996E-03 -6.97999999999999995E-01  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
PHIX           14    14     0(1P3E25.17)
  1.00000000535208861E+00 -4.95011076657771082E-01  1.45973361355003385E-06
  7.95948699576430917E-07  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.0000000000000000CE+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00 -1.03579331351297131E-08
  1.00014846463723353E+00 -5.93350047185587157E-06 -1.35234164642732054E-06
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  2.22771431496439276E-04  2.08467965555417058E-01
  9.93359001950856080E-01  2.84441802081675413E-02  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  1.55623923657108099E-02 -1.13088911441690967E+00  1.54843496026287862E-02
  9.92071235328456411E-01  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
```

```
  0.0000000000000000E+00  0.0000000000000000E+00  0.0000000000000000E+00
  9.9999913717409907E-01  1.1921793124392801E-03 -1.6952363206779723E-04
  2.6322903184384688E-05  0.0000000000000000E+00  0.0000000000000000E+00
  0.0000000000000000E+00  0.0000000000000000E+00  0.0000000000000000E+00
  0.0000000000000000E+00  0.0000000000000000E+00  0.0000000000000000E+00
  0.0000000000000000E+00  0.0000000000000000E+00 -2.1471072443918530E-03
  9.9758532774821413E-01 -2.8134634199282361E-01  4.3885784182855276E-02
  0.0000000000000000E+00  0.0000000000000000E+00  0.0000000000000000E+00
  0.0000000000000000E+00  0.0000000000000000E+00  0.0000000000000000E+00
  0.0000000000000000E+00  0.0000000000000000E+00  0.0000000000000000E+00
  0.0000000000000000E+00  1.5436429800994711E-02  2.7266471408372545E-03
  9.7607607272475972E-01 -1.3754004294272405E-03  0.0000000000000000E+00
  0.0000000000000000E+00  0.0000000000000000E+00  0.0000000000000000E+00
  0.0000000000000000E+00  0.0000000000000000E+00  0.0000000000000000E+00
  0.0000000000000000E+00  0.0000000000000000E+00  0.0000000000000000E+00
  2.8139348925562092E-03 -1.5293716134389612E-02  6.8152268781684332E-03
  9.9551958088738062E-01  0.0000000000000000E+00  0.0000000000000000E+00
  0.0000000000000000E+00  0.0000000000000000E+00  0.0000000000000000E+00
  0.0000000000000000E+00 -2.0048349031517501E-04  2.8857075477574031E-02
 -6.4898538871653376E-04 -2.4348097184355193E-02  5.0492406569182945E-04
 -6.0792658718130916E-05  5.9894164096110647E-02  7.0663043606749160E-03
  7.2933289872983760E-01  0.0000000000000000E+00  0.0000000000000000E+00
  0.0000000000000000E+00  0.0000000000000000E+00  0.0000000000000000E+00
 -2.0048349031517501E-04  2.8857075477574031E-02 -6.4898538871653376E-04
 -2.4348097184355193E-02 -5.0492406569182945E-04  6.0792658718130916E-05
 -5.9894164096110647E-02 -7.0663043606749160E-03  0.0000000000000000E+00
  7.2933289872983760E-01  0.0000000000000000E+00  0.0000000000000000E+00
  0.0000000000000000E+00  0.0000000000000000E+00  3.4659191709059983E-05
 -9.6128195579050024E-03 -3.9838307133742543E-04  4.2065581748140068E-03
  6.7898740942172157E-04  1.0712897995447662E-04  8.2031459295765437E-02
  9.7861071992968659E-04  0.0000000000000000E+00  0.0000000000000000E+00
  7.2933289872983760E-01  0.0000000000000000E+00  0.0000000000000000E+00
  0.0000000000000000E+00  3.4659191709059983E-05 -9.6128195579050024E-03
 -3.9838307133742543E-04  4.2065581748140068E-03 -6.7898740942172157E-04
 -1.0712897995447662E-04 -8.2031459295765437E-02 -9.7861071992968659E-04
  0.0000000000000000E+00  0.0000000000000000E+00  0.0000000000000000E+00
  7.2933289872983760E-01  0.0000000000000000E+00  0.0000000000000000E+00
  0.0000000000000000E+00  0.0000000000000000E+00  0.0000000000000000E+00
  0.0000000000000000E+00  2.8732070159447233E-04  4.0376134254462726E-04
  3.7505340801322658E-02 -1.5565279418756333E-02  0.0000000000000000E+00
  0.0000000000000000E+00  0.0000000000000000E+00  0.0000000000000000E+00
  7.2933289872983760E-01  0.0000000000000000E+00 -7.8306149914084954E-05
  2.5685222767435356E-02 -3.6681291156319174E-06 -9.6084634824104319E-03
  0.0000000000000000E+00  0.0000000000000000E+00  0.0000000000000000E+00
  0.0000000000000000E+00  0.0000000000000000E+00  0.0000000000000000E+00
  0.0000000000000000E+00  0.0000000000000000E+00  0.0000000000000000E+00
  7.7880078307140486E-01
BD             14     6     0(1P3E25.17)
 -3.1638800815363557E-05  4.5540072238046518E-03 -1.0241800665682798E-04
 -3.8424340869060539E-03  7.9683329116991835E-05 -9.5938414539550352E-06
  9.4520477714174614E-03  1.1151511569190101E-03  2.7291034808080250E-01
  0.0000000000000000E+00  0.0000000000000000E+00  0.0000000000000000E+00
  0.0000000000000000E+00  0.0000000000000000E+00 -3.1638800815363557E-05
  4.5540072238046518E-03 -1.0241800665682798E-04 -3.8424340869060539E-03
 -7.9683329116991835E-05  9.5938414539550352E-06 -9.4520477714174614E-03
 -1.1151511569190101E-03  0.0000000000000000E+00  2.7291034808080250E-01
  0.0000000000000000E+00  0.0000000000000000E+00  0.0000000000000000E+00
  0.0000000000000000E+00  5.4696536915860285E-06 -1.5170230864818832E-03
 -6.2869828445437453E-05  6.6384746196283546E-04  1.0715270054936543E-04
  1.6906292149065842E-05  1.2945589670112983E-02  1.5443700423890366E-04
  0.0000000000000000E+00  0.0000000000000000E+00  2.7291034808080250E-01
  0.0000000000000000E+00  0.0000000000000000E+00  0.0000000000000000E+00
  5.4696536915860285E-06 -1.5170230864818832E-03 -6.2869828445437453E-05
  6.6384746196283546E-04 -1.0715270054936543E-04 -1.6906292149065842E-05
 -1.2945589670112983E-02 -1.5443700423890366E-04  0.0000000000000000E+00
  0.0000000000000000E+00  0.0000000000000000E+00  2.7291034808080250E-01
```

```
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  4.53427982203776663E-05  6.37185868703239902E-05  5.91881159520873208E-03
 -2.45639565827248394E-03  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  2.72910348080802501E-01
  0.00000000000000000E+00 -9.78826873926061928E-06  3.21065284592941959E-03
 -4.58516139453989684E-07 -1.20105793530130399E-03  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  2.22350097883925610E-01
CQD                 8    8     0(1P3E25.17)
  4.44126633999966317E-07  3.19824114002613725E-04 -7.73825275016912623E-06
  2.81946508621761111E-05  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  1.12312975915254400E-03 -1.72813321285079853E-05  9.03053221962999628E-06
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  7.51338225803705795E-06 -1.06394712062323290E-05  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  2.50301658324871191E-05  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  4.22118869715917454E-06  2.35533551754844802E-06  2.74033425769201685E-04
 -4.25622104595152692E-05  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  1.98632296576301042E-05  1.31344577366704238E-05 -7.53119316798235158E-07
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  2.81031967997728679E-04 -4.24966227594259779E-05  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  1.07442557043893908E-05
H                   7   14     0(1P3E25.17)
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  1.08833893978467892E-05  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  1.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  4.89982132195794799E-03  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  1.00000000000000000E+00  0.00000000000000000E+00  5.67704252850563296E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  1.00000000000000000E+00
  2.31488449786658190E-02  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  9.83960432316528785E-01  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00 -1.39460423034596978E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  1.00000000000000000E+00  0.00000000000000000E+00
  2.26038242783957138E+00  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  1.00000000000000000E+00  2.26502498830811114E-01  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  4.33479035363193196E-01
  0.00000000000000000E+00  0.00000000000000000E+00 -8.70878517508958186E-02
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  4.33479035363193196E-01  0.00000000000000000E+00  0.00000000000000000E+00
  8.70878517508958186E-02  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  4.17719882519488814E-01  0.00000000000000000E+00
  0.00000000000000000E+00 -3.73973353427373601E-03  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  4.17719882519488814E-01
  0.00000000000000000E+00  0.00000000000000000E+00  3.73973353427373601E-03
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  2.15125329664152855E-01  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00 -6.95627329192546572E-02  0.00000000000000000E+00
```

```
  0.00000000000000000E+00  0.00000000000000000E+00
GRP             14    7    0(1P3E25.17)
 -5.68696920090312372E-03  4.31759862632571087E-01 -6.23040722973446082E-03
 -1.10478065321356321E-02 -3.33173879019916454E-17  1.08679352388865141E-18
  7.50726633824691472E-18 -1.80423813065778143E-17  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00 -3.99140534364858416E-04
 -1.86912216892033502E-03  8.31285832101947005E-04  3.03096918442494231E-04
  8.11771906417327951E-18 -3.30162588467134283E-18 -3.27554612555877601E-19
  5.54155808505053480E-18  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  2.50446155229867115E-04 -1.47304087095141763E-03
  1.34709741529997457E-04  1.38253345507315461E-03  3.42661194013248612E-18
 -6.14120235779420291E-19 -7.50525023375453795E-19  2.41101480972898245E-18
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
 -3.61798810269901767E-04 -1.60850623135131546E-03  7.54735785137120469E-04
  2.84235146256431498E-04  7.39426078821427279E-18 -3.00381699283074801E-18
 -3.02015569487093676E-19  5.04938994161766352E-18  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00 -1.26788905188511254E-19
  5.23386446650688831E-20  1.10455114782422435E-19  6.24976421873315982E-20
  1.05533972772336339E-05 -6.16557106929206699E-05  1.83666037948018786E-03
 -2.86782141150179320E-04  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  2.78934862576855574E-18  1.23250623774500211E-18
 -1.92205928904073802E-18 -1.55915287663957437E-18 -2.46393191311634859E-04
 -2.74261821204954849E-04 -3.18646823500198887E-03  9.74311808709009118E-04
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  1.70979278620633204E-18  3.65605425838873679E-18 -1.43316009722296882E-18
 -2.59022953168947215E-18  2.24705981030346884E-03 -3.81013533198324008E-03
  6.69274786624041388E-02 -9.85234636932846614E-03  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00
R                7    7    0(1P3E25.17)
  4.79999999999999999E-06  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  1.60000000000000001E-05
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  3.60000000000000001E-05  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  9.99999999999999997E-05  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  3.99999999999999999E-04
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  3.60000000000000001E-05  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  0.00000000000000000E+00  0.00000000000000000E+00  0.00000000000000000E+00
  2.49999999999999999E-05
AR               7    7    0(1P3E25.17)
  8.44835530065477312E-06 -5.29081162715315622E-08 -9.35620919069142460E-08
 -2.84651713248438937E-07  4.85708244071860591E-23  7.77671234801839623E-23
  1.91377961886230228E-22 -5.29081162715316540E-08  1.60137009421948264E-05
  5.46785184789202375E-09  7.76480990794858696E-08  4.15995364098802136E-23
 -6.97116655955456029E-23 -4.31393022768572762E-23 -9.35620919069142460E-08
  5.46785184789202209E-09  3.60508858965642217E-05  3.17608435844447582E-08
  1.96814631551441122E-23 -5.65094267891905874E-23 -7.85334530686311817E-23
 -2.84651713248439460E-07  7.76480990794858696E-08  3.17608435844447673E-08
  1.00440151666916966E-04  2.36855404569113622E-22 -3.96682087611602476E-22
 -2.45783365333836221E-22  4.95690141866152616E-23  1.08460237189547690E-23
 -1.91397239144710821E-23  6.13731288366493682E-23  4.00869823489018065E-04
 -1.34691889620975557E-07  1.98920649813100631E-06 -1.52397213645011041E-22
```

```
  8.77135103286168789E-23  8.75719494582949750E-23  4.99233330219296148E-22
 -1.34691889620975709E-07  3.60380377524117259E-05 -2.93010348005608602E-07
 -2.18224412905572546E-22  2.50655060210600563E-22  1.33657288229167230E-22
  1.42500386131614850E-21  1.98920649813100636E-06 -2.93010348005608225E-07
  2.96532810269704434E-05                        .
AKINV            7    7     0(1P3E25.17)
  1.18383361951547689E+05  3.89400451858403130E+02  3.06883514781545351E+02
  3.35105464864857382E+02 -1.09038843052226842E-14 -2.56772132863542110E-13
 -7.61677970497653486E-13  3.89400451858403812E+02  6.24480446354936284E+04
 -8.41935884562484094E+00 -4.71709865710700287E+01 -6.90344467390140239E-15
  1.20128705565046127E-13  8.95725060764355498E-14  3.06883514781545344E+02
 -8.41935884562483983E+00  2.77393740706924127E+04 -7.89542072934531947E+00
 -1.73604561631476669E-15  4.33098021288770658E-14  7.19508203247075632E-14
  3.35105464864858000E+02 -4.71709865710700287E+01 -7.89542072934532047E+00
  9.95716638465379515E+03 -6.28760202455375862E-15  1.09416354965225313E-13
  8.17813902303013309E-14 -1.87681658456172870E-14  8.18886531389694017E-16
  1.87631255843863453E-15  7.55038923717734193E-16  2.49540834905129952E+03
  7.96617058750497242E+00 -1.67318696656010349E+02  5.01177258516050393E-13
 -1.53932169029181521E-13 -6.69726336035828468E-14 -1.40260831711601760E-13
  7.96617058750498108E+00  2.77507135608691938E+04  2.73676288036901845E+02
  8.56639591533689699E-13 -5.24270153955656154E-13 -1.23109301975309289E-13
 -4.77032987214572816E-13 -1.67318696656010353E+02  2.73676288036901489E+02
  3.37370094227732652E+04
DETAK            1    1     0(1P3E25.17)
  2.09739364265554114E-31
```

APPENDIX F: NAECON PAPER

This appendix presents a paper submitted for the IEEE National Aerospace and Electronics Conference (NAECON) conference on May 20, 1992 in Dayton, OH. This paper is referenced in Chapter 4. The paper details some single-failure data. The primary contribution of the data within the paper to this thesis is the investigation and development of the dither signals used throughout this thesis effort.

MULTIPLE MODEL ADAPTIVE ESTIMATION APPLIED TO THE VISTA F-16 FLIGHT CONTROL SYSTEM WITH ACTUATOR AND SENSOR FAILURES

Timothy E. Menke and Peter S. Maybeck

Department of Electrical and Computer Engineering
Air Force Institute of Technology / ENG
Wright-Patterson AFB, Ohio 45433

ABSTRACT

Multiple model estimation (MMAE) is applied to the Variable In-flight Stability Test Aircraft (VISTA) F-16 flight control system. Single actuator and hard sensor failures are introduced and system performance is evaluated. Performance is enhanced by the application of a modified Bayesian form of MMAE, scalar residual monitoring to reduce ambiguities, dithering where advantageous, and purposeful commands.

1. Introduction

For many applications, it is highly desirable to develop an aircraft flight control system with reconfigurable capabilities: able to detect and isolate failures of sensors and/or actuators and then to employ a control algorithm that has been specifically designed for the current failure mode status. One means of accomplishing this, in a manner that is ideally suited to distributive computation, is multiple model adaptive estimation (MMAE) [1-4] and control (MMAC) [5-7].

Assume that the aircraft system is adequately represented by a linear perturbation stochastic state model, with a (failure status) uncertain parameter vector affecting the matrices defining the structure of the model or depicting the statistics of the noises entering it. Further assume that the parameters can take on only discrete values: either this is reasonable physically (as for many failure detection formulations), or representative discrete values are chosen throughout the continuous range of possible values. Then a Kalman filter is designed for each choice of parameter value, resulting in a bank of K separate "elemental" filters. Based upon the observed characteristics of the residuals in these K filters, the conditional probabilities of each discrete parameter value being "correct", given the measurement history to that time, are evaluated iteratively. In MMAC, a separate set of controller gains is associated with each elemental filter in the bank. The control value of each elemental controller is weighted by its corresponding probability, and the adaptive control is produced as the probability weighted average of the elemental controller outputs. As one alternative (using maximum a posteriori, or MAP, rather than minimum mean square error, or MMSE, criteria for optimality), the control value from the single elemental controller associated with the highest conditional probability can be selected as the output of the adaptive controller.

Previous efforts investigated the application of a multiple model adaptive control algorithm to a short takeoff and landing (STOL) F-15 [8,9]. The system was modeled with four elemental controllers designed for a healthy aircraft, failed pitch rate sensor, failed stabilator, or failed "pseudo-surface" - a combination of canards, ailerons, and trailing edge flaps. Conclusions from this study indicated that the elemental filters must be carefully tuned to avoid masking of "good" versus "bad" models. This observation is not compatible with Loop Transmission Recovery (LTR) tuning techniques. Other research efforts demonstrated the effectiveness of the MMAC algorithm using seven elemental controllers designed for a healthy aircraft, one of three actuator failures, or one of three sensor failures [10,11]. The study included effects of single and double failures, and partial failures as well as hard failures. It also demonstrated the effectiveness of alternate techniques to resolve ambiguities using modified computational techniques and scalar residual monitoring.

2. MMAC and MMAE-Based Control

Let a denote the vector of uncertain parameters in a given linear stochastic state model for a dynamic system, in this case depicting the failure status of sensors and actuators of the aircraft. These parameters can affect the matrices defining the structure of the model or depicting the statistics of the noises entering it. In order to make simultaneous estimation of states and parameters tractable, it is assumed that a can take on only one of K discrete representative values. If we define the hypothesis conditional probability $p_k(t_i)$ as the probability that a assumes the value a_k (for $k = 1,2,\ldots,K$), conditioned on the observed measurement history to time t_i:

$$p_k(t_i) = Prob[a = a_k \mid Z(t_i) = Z_i] \qquad (1)$$

then it can be shown [1-4] that $p_k(t_i)$ can be evaluated recursively for all k via the iteration:

$$p_k(t_i) = \frac{f_{z(t_i)|a,z(t_{i-1})}(z_i|a_k, Z_{i-1})\, p_k(t_{i-1})}{\sum_{j=1}^{K} f_{z(t_i)|a,z(t_{i-1})}(z_i|a_j, Z_{i-1})\, p_j(t_{i-1})} \quad (2)$$

in terms of the previous values of $p_1(t_{i-1}), \ldots, p_K(t_{i-1})$, and conditional probability densities for the current measurement $z(t_i)$ to be defined explicitly in Equation (12). Notionally, the measurement history random vector $Z(t_i)$ is made up of partitions $z(t_1), \ldots, z(t_i)$ that are the measurements available at the sample times $t_1, \ldots, t_i$; similarly, the realization Z_i of the measurement history vector has partitions $z_1, \ldots, z_i$. Furthermore, the Bayesian multiple model adaptive controller output is the probability weighted average [5-7]:

$$u_{MMAC}(t_i) = \sum_{k=1}^{K} u_k[\hat{x}_k(t_i^+), t_i]\cdot p_k(t_i) \quad (3)$$

Here $u_k[\hat{x}(t_i), t_i]$ is a deterministic optimal full-state feedback control law based on the assumption that the parameter vector equals a_k, and $\hat{x}_k(t_i^+)$ is the state estimate generated by a Kalman filter similarly based on the assumption that $a = a_k$. If the parameter were in fact equal to a_k, then certainty equivalence [5] would allow the LQG (Linear system, Quadratic cost, Gaussian noise) optimal stochastic control to be generated as one of the $u_k[\hat{x}_k(t_k^+), t_i]$ terms in the summation of Eq. (3).

More explicitly, let the model corresponding to a_k be described by an "equivalent discrete-time model [4,5,11] for a continuous-time system with sampled data measurements:

$$x_k(t_{i+1}) = \Phi_k(t_{i+1}, t_i)x_k(t_i) + B_k(t_i)u(t_i) + G_k(t_i)w_k(t_i) \quad (4)$$

$$z(t_i) = H_k(t_i)x_k(t_i) + v_k(t_i) \quad (5)$$

where x_k is the state, u is a control input, w_k is discrete-time zero-mean white Gaussian dynamics noise of covariance $Q_k(t_i)$ at each t_i, z is the measurement vector, and v_k is discrete-time zero-mean white Gaussian measurement noise of covariance $R_k(t_i)$ at t_i, assumed independent of w_k; the initial state $x(t_o)$ is modeled as Gaussian, with mean x_{ko} and covariance P_{ko} and is assumed independent of w_k and v_k. Based on this model, the Kalman filter [11] is specified by the measurement update:

$$A_k(t_i) = H_k(t_i)P_k(t_i^-)H_k^T(t_i) + R_k(t_i) \quad (6)$$

$$K_k(t_i) = P_k(t_i^-)H_k^T(t_i)A_k^{-1}(t_i) \quad (7)$$

$$\hat{x}(t_i^+) = \hat{x}_k(t_i^-) + K_k(t_i)\cdot[z_i - H_k(t_i)\hat{x}_k(t_i^-)] \quad (8)$$

$$P_k(t_i^+) = P_k(t_i^-) - K_k(t_i)H_k(t_i)P_k(t_i^-) \quad (9)$$

and the propagation relation:

$$\hat{x}_k(t_{i+1}^-) = \Phi_k(t_{i+1}, t_i)\hat{x}_k(t_i^+) + B_k(t_i)u(t_i) \quad (10)$$

$$P_k(t_{i+1}^-) = \Phi_k(t_{i+1}, t_i)P_k(t_i^+)\Phi_k^T(t_{i+1}, t_i) + G_k(t_i)Q_k(t_i)G_k^T(t_i) \quad (11)$$

The multiple model adaptive estimation (MMAE) algorithm is composed of a bank of K separate Kalman filters, each based on a particular value $a_1, \ldots, a_K$ of the parameter vector, as depicted in Figure 1. Instead of generating a control vector u_k, the MMAE generates a probabilistically weighted state estimate vector, $x_{MMAE}(t_i)$. These state estimates are used by the flight control system to generate the control vector u_k. Such MMAE-based control is used in this research rather than a MMAC because the incorporation of the full VISTA F-16 flight control system illustrates another step toward the maturation of the MMAC/MMAE algorithms. When the measurement z_i becomes available at time t_i, the residual vector r_k is generated in each of the K filters according to the bracketed term in Eq. (8), and used to compute $p_1(t_i), \ldots, p_K(t_i)$ via Eq. (2). Each numerator density function in (2) is given by the Gaussian form:

$$f_{z(t_i)|a,z(t_{i-1})}(z_i|a_k, Z_{i-1}) = \frac{1}{(2\pi)^{(m/2)}|A_k(t_i)|^{1/2}}\exp\{\cdot\} \quad (12)$$

$$\{\cdot\} = [-\frac{1}{2}r_k^T(t_i)A_K^{-1}(t_i)r_k(t_i)] \quad (13)$$

where m is the measurement dimension and $A_k(t_i)$ is calculated in the k-th Kalman filter as in Eq. (6). The denominator in Eq. (2) is simply the sum of all the computed numerator terms and thus is the scale factor required to ensure that the $p_k(t_i)$'s sum to one.

One expects the residuals of the Kalman filter based upon the "best" model to have mean squared value most in consonance with its own computed $A_k(t_i)$, while "mismatched" filters will have larger residuals than anticipated through $A_k(t_i)$. Therefore, Eqs. (2), (3), and (6) - (12) will most heavily weight the filter based upon the most correct assumed parameter value. However, the performance of the algorithm depends on there being significant differences in the characteristics of residuals in "correct" vs. "mismatched" filters. Each filter should be tuned for best performance when the "true" values of the uncertain parameters are identical to its assumed value for these parameters. One should specifically avoid the "conservative" philosophy of adding considerable dynamics pseudonoise, often used to guard against

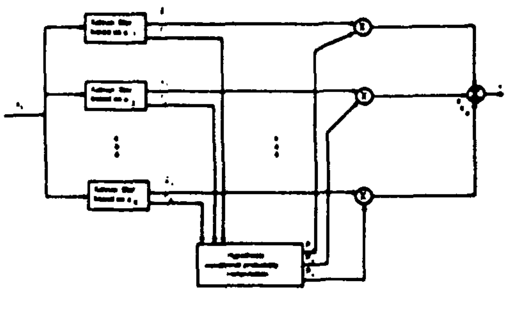


Figure 1. Multiple Model Adaptive Estimation Algorithm

divergence, since this tends to mask the differences between good and bad models.

3. System Modeling

Aerodynamic Model. A six-degree-of-freedom nonlinear aerodynamic model provided data to generate a linearized perturbation model utilized in this study. The data base resides within the Flight Dynamics Laboratory at Wright-Patterson AFB, Ohio. The linearized model includes increments for pitch attitude, pitch rate, angle of attack, velocity, roll angle, sideslip angle, roll rate, and yaw rate. Normal and lateral accelerations are computed. Control effects are given by left and right stabilators, left and right flaperons, rudder and leading edge flaps. The model is developed with constant thrust.

Flight Control System. The flight control system (FCS) model is a Fortran representation of the VISTA F-16 FCS. The model accurately depicts the true system by including longitudinal, lateral, and directional channels. Each channel provides command force gradients, command limiting, signal magnitude and rate limiting accomplished within the controller software, gain scheduling, biases, filtering characteristics, and true surface position and rate limiting. Sensor measurements are corrected for position error where applicable. The flight control system requires seven sensor inputs for proper performance including: velocity, angle of attack, pitch rate, normal acceleration, roll rate, yaw rate, and lateral acceleration.

The development of a detailed model allows for a realistic evaluation of the MMAE algorithm. The flight control system and linearized aerodynamic models were validated separately and as a system using a six-degree-of-freedom nonlinear simulation. Results indicated excellent correlation provided that the constraints of the linear aerodynamic perturbation model were not violated. Given the short convergence times typical for a fault detection and isolation algorithm, this is not a restrictive constraint.

4. Algorithm Implementation

Hypothesized Failures. The parameter space, denoted by the vector quantity a, was descretized into twelve hypothesized hard failures: left stabilator, right stabilator, left flaperon, right flaperon, rudder, velocity sensor, angle of attack sensor, pitch rate sensor, normal acceleration sensor, roll rate sensor, yaw rate sensor, and the lateral acceleration sensor. Additionally, the no-failure aircraft condition was included to provide an initial system configuration prior to failure transition. Total or "hard" actuator failures are modeled by zeroing out the appropriate columns of the control input matrix B of Eq. (4) and hard sensor failures are modeled by zeroing out the corresponding rows of the measurement matrix H of Eq. (5).

Bayesian Form. The final probability-weighted average of the state estimates, computed as shown in Figure 1, is produced by a Bayesian form of the MMAE algorithm. A Bayesian form of the MMAE algorithm allows for a blending of filters designed for hard failures and those designed for no-failures to address partial or soft failures. Practical implementation requires a lower bound when computing the probabilities according to Eq. (2). The addition of a lower bound prevents the algorithm from assigning any single $p_k(t_i)$ a value of zero, which would prevent it from being considered in future probability computations. From the iterative nature of Eq. (2), if $p_k(t_{i-1})$ were assigned a value of zero for one of the filters, subsequent probability calculations for that filter would also assign a probability of zero (i.e. $p_k(t_i) = 0$). The addition of a lower bound provides another favorable characteristic. The number of iterations required to increase a very small, but nonzero, p_k is directly proportional to the magnitude of the p_k. By providing a lower bound we allow p_k values, previously not important to the combined state estimate, to increase in a timely manner if the system state changes.

"Beta Dominance". As discussed earlier in Section 2, the hypothesis probabilities $p_k(t_i)$ are calculated according to Eq. (2). Earlier efforts [2,4,6,10] noted that the leading coefficient preceding the exponential term in Eq. (12) does not provide any useful information in the identification of the failure. As discussed in Section 2, the likelihood quotient,

$$L_k(t_i) = r_k^T(t_i) A_k^{-1} r_k(t_i) \tag{14}$$

compares the residual with the hypothesized filter's internally computed residual covariance. Filters with residuals that have mean square values most in consonance with their internally computed covariance are assigned the higher probabilities by the MMAE algorithm. However, if the likelihood quotients were nearly identical in magnitude for all k, the probability computations would be based upon the magnitude of the determinants of the $A_k(t_i)$ matrices, resulting in an incorrect assignment of the probabilities. This effect is known as "Beta Dominance". Because sensor failures, as simulated by zeroing out a row of H, yield

smaller $A_k(t_i)$ values, "beta dominance" produces a tendency to generate false alarms about sensor failures.

Previous efforts removed the term preceding the exponential in Eq. (12). Since the denominator of Eq. (2) contains the summation of all numerator terms, excluding the terms preceding the exponentials in the calculation of the probabilities does not alter the fact that the computed probabilities sum to one.

Scalar Residual Monitoring. Incorrect or ambiguous failure identification may be resolved through the use of scalar residual monitoring. Eqs. (2), (12), and (13) demonstrate the relationship of the probability calculations, the probability density function, and the likelihood quotient. These three equations demonstrate the dependency of the probability calculations on the magnitude of the likelihood quotient, Eq. (13). The likelihood quotient is merely the sum of scalar terms relating the product of any two scalar components of the residual vector and the filter's internally computed covariance for those two components. If a sensor failure occurs, the single scalar term associated solely with that sensor should have a residual value whose magnitude is much larger than the associated variance in all of the elemental filters except for the filter designed to "look" for that sensor failure. Scalar residual monitoring can be used as an additional vote when attempting to reduce or eliminate failure identification ambiguities.

Purposeful Commands. Failure detection and isolation using the MMAE algorithm requires a stimulus to disturb the system from a quiescent state. The MMAE algorithm's performance depends upon the magnitude of the residuals within incorrect filters having large residual values. Residuals are the difference between measurements and filter predictions of those measurements. Incorrect filters will provide poor estimates relative to the filter based on the "true" system status. Small deviations from a quiescent state will be virtually indistinguishable from system noise, providing poor failure detection and identification. Having justified the need for stimuli to "shake up" the system, rationale was developed to select stimuli, control deflections, and improve performance. Previous efforts selected a pitch down maneuver to aid in the identification process for the longitudinal axis of an aircraft with generally favorable results [14]. However, fundamental differences exist between earlier research and this effort. Earlier efforts concentrated on applying the MMAC algorithm, evaluating its performance, and designing algorithms to maintain stability and control in the longitudinal axis. A longitudinal pitch down maneuver was sufficient to provide enough system excitation for good performance. A three-axis sophisticated control system requires excitation in multiple axes to provide adequate residual growth in filters whose hypothesis does not reflect the true system failure status. The purposeful commands used in this effort were longitudinal stick pulls, lateral stick pulses, and varying amounts of rudder application. Ordinary aircraft maneuvering would probably be more than sufficient to provide adequate excitation and good performance; straight-and-level flight would be more challenging (though less flight critical) for a failure detection system.

Autonomous Dithering. Autonomous dithering enhances failure detection and identification by providing sufficient excitation in benign non-maneuvering flight conditions or as a pilot-selectable option. A number of dither signals were evaluated, including square waves, triangle waves, combinations of these forms, and sine waves. Pulse trains using a square wave form produced good performance with one drawback, failures are not detected until the application of the pulse. Additionally, pilots may find the application of a dither signal of sufficient strength to provide good failure detection and isolation objectionable, unless he were able to turn such dithering on and off himself. Sufficient data was not available to relate pilot comments and normal and lateral accelerations in this application, so dithers were designed to be as subliminal as possible while yielding desired identifiability of failures.

5. Performance

The application of the multiple model adaptive estimation algorithms to the VISTA F-16 aircraft in a low dynamic pressure case provided an interesting test for this technique. The flight condition, 0.4 Mach at an altitude of 20000 ft., demonstrated algorithm performance in a low dynamic pressure scenario. Earlier efforts studied the VISTA F-16 at a higher dynamic pressure and emphasized different failure scenarios and characteristics [15]; the case of low dynamic pressure yields a more difficult identification problem. The original goal was to evaluate the MMAE algorithm's ability to detect and isolate failures within the flight control system and not to evaluate the ability of the controller to maintain control of the vehicle after the identification of the failure. An added benefit of using the VISTA F-16 flight control system was the absence of any single-failure-induced loss of control. The figures presented in this section are single data runs as opposed to Monte Carlo runs averaged over a number of runs, in order to exhibit real-time signal characteristics (Monte Carlo runs were used to corroborate performance attributes over multiple experimental trials).

Purposeful Commands. Figure 2 demonstrates a left stabilator failure induced at 3.0 seconds. A square wave dither signal occurs in all three channels every 3.0 seconds beginning at 0 seconds. The pulse widths and magnitudes were different for each channel and were determined by trial and error. Typical pulse cycles, the application of a pulse of positive amplitude followed by the application of a pulse of negative amplitude, were usually approximately 0.25 seconds. Figure 2 presents performance data after a failure at 3.0 seconds and the application of a longitudinal stick pull for a duration of 3.0 seconds, starting at 3.0 seconds. It displays only the no-failure and failed actuator elemental filter probabilities; the failed-sensor elemental filters never

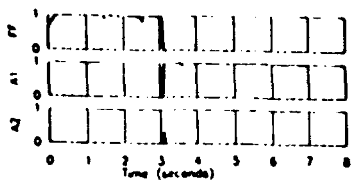


Figure 2. Probability for a Left Stabilator Failure Using a Purposeful Command

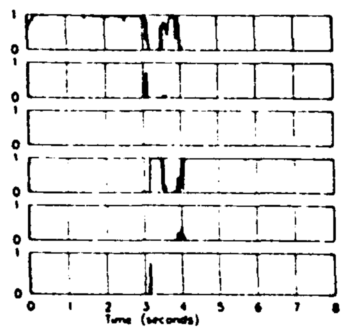


Figure 3. Probability for a Left Flaperon Failure Using a Purposeful Command

attained any appreciable portion of the total probability. The FF, A1, A2, A3, A4, and A5 designations are the fully functional (no failure), left stabilator, right stabilator, left flaperon, right flaperon, and rudder elemental filters, respectively. For the left stabilator failure with a simultaneous purposeful command of 10 lbs aft stick, the algorithm exhibits a lag time of approximately 0.2 seconds prior to positive failure identification. A small spike is evident in the right stabilator elemental filter during the detection and decision period. Occasionally, ambiguities arise between the left and right stabilator for small periods of time during a stabilator failure (purposeful roll rates could be used to isolate which stabilator failed, once the algorithm detects that one of the stabilators has failed). Since the left and right stabilators provide pitch control and augment the roll channel, the identification task is significantly more difficult than that of an actuator dedicated to a single channel task. If the aircraft has a roll angle, the surface positions of the left and right actuator may not be the same. If one of the surface positions is smaller than the other, failing one surface may produce a different system response from failing the other. The result may provide different probability convergence phenomena. The solution is usually to increase the purposeful command or dither signal to a level sufficient to produce proper system excitation. However, if the dither command is too large, a pilot may object to normal or lateral accelerations that result from commands which he did not initiate. A large purposeful or dither command may reduce the algorithm performance by inducing large transients. The Kalman filter gains within each of the elemental filters are designed for steady state performance. The system state variables will require a longer settling time as larger amplitude transients are produced. Increased stick activity can produce the same effect.

Figure 3 illustrates a left flaperon failure induced at 3.0 seconds. In this failure scenario, a rudder application of 45 lbs for a duration of 3.0 seconds combined with a longitudinal and lateral stick pulse demonstrate algorithm performance. The flaperons are control surfaces which do not produce significant changes in state parameters in a short period of time. In this case, the failure detection is identified in approximately 0.2 seconds. Thereafter, the algorithm attempts to declare a fully functional aircraft, and finally 0.6 seconds later, positively identifies that failure as a failed left flaperon. During the 0.6 second interval when the left flaperon is not selected, four filters share the total probability, including: fully functional, left flaperon, right flaperon, and the pitch rate sensor (not shown). If a lateral stick command is introduced from 3 sec to 6 sec of the simulation, the probability remains at the fully functional filter until approximately 3.8 seconds. At that time, the probability is transferred directly and entirely to the left flaperon and remains there for the duration of the run.

Figure 4 shows the performance of the algorithm for a rudder failure induced at 3.0 seconds. Control applications are given by a 45 lb rudder kick and hold for a duration of 3.0 seconds, and a longitudinal and lateral stick pulse. Results demonstrate a 0.2 second lag between the induction of the failure and positive identification by the proper elemental filter. The "drop out" of the probability during 3.6 to 3.8 second interval is gained by the yaw rate sensor elemental filter (not shown).

Figure 5 depicts the elemental filter probabilities for the seven elemental filters that assume failed sensors. A pitch rate failure is induced at 3.0 seconds while simultaneously applying a longitudinal command of 10 lbs aft stick. The labels S1, S2, S3, S4, S5, S6, and S7 are the sensor designations for the velocity, angle of attack, pitch rate, normal acceleration, roll rate, yaw rate, and lateral acceleration elemental filters, respectively. In this scenario, the probability is transferred directly from the fully functional filter (not shown) to the pitch rate sensor filter, S3, at the time of the failure. The lag to failure detection and identification in this case is less than 0.2 seconds. Sensor failures are usually identified quickly due to the direct relationship between the variable the sensor measures and the residual calculation upon which the probabilities are based.

In general, purposeful commands aid in the

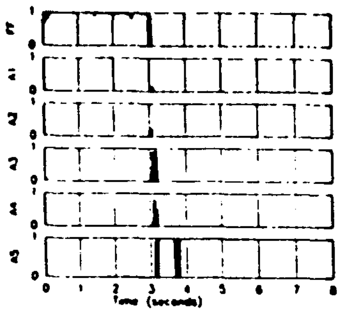


Figure 4. Probability for a Rudder Failure Using a Purposeful Command

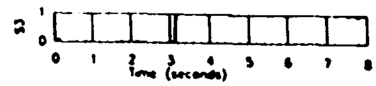


Figure 5. Probability for a Pitch Rate Sensor Failure Using a Purposeful Command

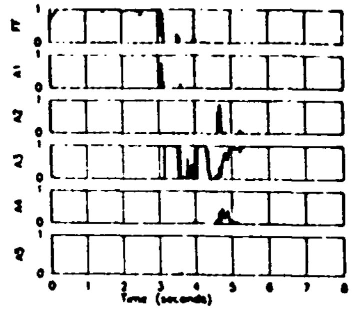


Figure 6. Probability for a Flaperon Failure Using a Subliminal Dither

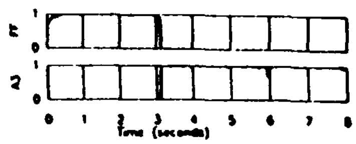


Figure 7. Probability for a Flaperon Failure Using a Large Amplitude Dither

identification process and often enhance performance. However, periods of large amplitude or high frequency stick activity can cause ambiguities and delay the identification process. Each axis must be stimulated by a control input to achieve good performance. Typical flight control maneuvers should be more than sufficient to provide the level of excitation necessary to achieve acceptable algorithm performance. Dither signals optimized to provide good failure detection and identification characteristics can provide the best algorithm performance, when used to augment typical maneuver inputs (i.e., dither is added to a particular channel if the input commands do not excite that channel).

Identification of Failure in Benign Flight Conditions. For flight conditions where little control activity is present, flight safety can be maintained through the use of autonomous dithering signals, or pulses. As previously described in this section, a dither signal is applied to each axis every 3.0 seconds. Dither signal amplitudes and frequencies were artificially limited to produce no more than +/- 0.05 g's normal acceleration and +/- 0.1 g's lateral acceleration. These restrictions were developed to allow a dither system to run in the "background" during the flight phase, providing failure detection capability in benign flight conditions. The dither was temporarily disabled when a pilot command was induced in that channel. Dither commands in channels without a pilot command were executed.

Figure 6 illustrates a left flaperon failure induced at 3.0 seconds. In this case, the failure is detected initially but not locked until approximately 4.9 seconds. The "missing" probability was picked up by the yaw rate filter and the lateral acceleration filter (not shown). In this scenario, performance could be enhanced by increasing the pulse amplitude of the dither signal. Figure 7 doubled the rudder pulse amplitude. The correct failure was identified in approximately 0.16 seconds. The lateral acceleration was approximately 0.2 g's, probably too large to be undetected by a pilot. While this dither may be unacceptable as a "background" dither, it is perfectly acceptable as a failure identification test. If a pilot believed a failure existed but could not identify the failure, he would select this option.

Figure 8 displays a rudder failure induced at 3.0 seconds. The dither signal was not large enough to affect immediate identification. The correct failure is identified after a delay of approximately 2.2 seconds. The angle of attack, pitch rate, normal acceleration, roll rate, and yaw rate sensor all contain some portion of the probability throughout the 8-second run. This suggests insufficient excitation to provide good algorithm performance. Figure 9 increases the amplitude of the rudder dither pulse. In this case, the rudder failure is identified after a delay of approximately 0.1 second. The notch in the probability at approximately 6.0 seconds is due to the application of another dither pulse. This pulse shakes up the system to enhance

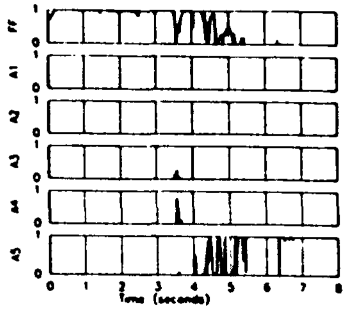


Figure 8. Probability for a Rudder Failure Using a Subliminal Dither

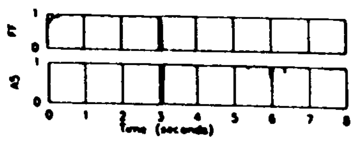


Figure 9. Probability for a Rudder Failure Using a Large Amplitude Dither

identifiability. However, the Kalman filters were designed using steady state gains. After the application of the dither pulses, the system returns to a steady state condition and again the rudder is identified as the correct failure.

Residual Characteristics. Figure 10 illustrates residual characteristics for a left stabilator failure. Figure 10 is the velocity residual for the elemental filter assuming a left stabilator has failed. The velocity residual was selected for display since it provides clear indications that a failure has occurred. In this scenario, a constantly applied sine wave dither signal was developed using the normal and lateral acceleration criteria discussed earlier. The frequency of the dither was approximately 2.38 Hz. Prior to the induction of the failure, the residual violated the 2 sigma bounds (+/- 0.0058 ft/sec), appeared time correlated rather than white, and the residual frequency matched the dither frequency. The 2 sigma bound is based on the left stabilator elemental filter's internally computed variance for the velocity residual. This behavior clearly indicates that the hypothesis of a failed left stabilator is incorrect for the first 3 sec. of the simulation. After the declaration of a left stabilator failure at 3.0 seconds, the velocity residual appears more white and moves within the 2 sigma bounds; note, however, that it takes about a second for the apparent residual

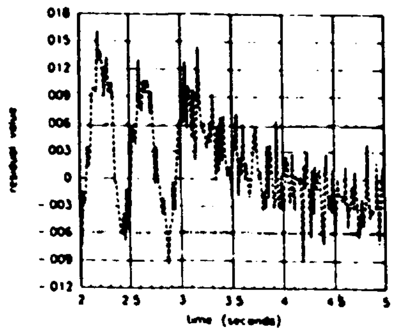


Figure 10. Velocity Residual Characteristics for a Left Stabilator Failure

bias to reduce to a negligible value. Scalar residual monitoring provides positive evidence of a failure. This additional voter is useful in the reduction of ambiguities in actuator failures. For actuator failures, initial results indicate that the velocity, normal acceleration, yaw rate, and lateral acceleration residuals provide the best indications of a failure.

6. Summary

A multiple model adaptive estimation algorithm with one fully functional, five failed-actuator and seven failed-sensor elemental filters illustrates the algorithm's performance when applied to a VISTA F-16 flight control system using a linearized aerodynamic model. A modified Bayesian approach allows for a blending of state estimates and provides lower bounds to enhance algorithm convergence properties. Compensation for "Beta Dominance" enhances algorithm performance by not allowing the term preceding the exponentiation in Eq. (12) to enter into the calculations. This term biases the calculation of the probabilities toward the filter whose $A_k(t_i)$ matrices have the smallest determinants. Scalar residual monitoring aids in resolving ambiguities by demonstrating residual characteristics consistent with a true failure.

The algorithm demonstrates good convergence characteristics during purposeful commands and dither signals. Optimizing the dither to improve algorithm performance is effective. However, large dither signals cannot be considered subliminal and may be considered objectionable by a pilot; allowing him to turn the dither on and off may be more useful practically.

References

1. Magill, D.T., "Optimal Adaptive Estimation of Sampled Stochastic Processes," IEEE Trans AC, Vol. AC-10, No. 5, pp. 434-439, Oct. 1965.

2. Athans, M., and C.B. Chang, "Adaptive Estimation and Parameter Identification Using Multiple Model Estimation Algorithm," Technical Note 1976-28, ESD-TR-76-184, Lincoln Laboratory, Lexington, Mass., June 1976.

3. Maybeck, P.S., Stochastic Models, Estimation and Control, Vol. 1, Academic Press, New York, 1979.

4. Maybeck, P.S., Stochastic Models, Estimation and Control, Vol. 2, Academic Press, New York, 1982.

5. Maybeck, P.S., Stochastic Models, Estimation and Control, Vol. 3, Academic Press, New York, 1982.

6. Athans, M., et. al., "The Stochastic Control of the F-8C Aircraft Using a Multiple Model Adaptive Control (MMAC) Method - Part 1: Equilibrium Flight," IEEE Trans. AC, Vol AC-22, No. 5, pp. 768-780, Oct. 1977.

7. Greene, C.S., and A.S. Willsky, "An Analysis of the Multiple Model Adaptive Control Algorithm," Proc IEEE Conf. Dec. and Cont., Albuquerque, New Mexico, pp. 1142-1145, Dec. 1980.

8. Pogoda, D.L., "Multiple Model Adaptive Controller for the STOL F-15 with Sensor/Actuator Failures," M.S. thesis, A.F. Inst. of Tech., Wright-Patterson AFB, Ohio, Dec. 1988.

9. Maybeck, P.S., and D.L. Pogoda, "Multiple Model Adaptive Controller for the STOL F-15 with Sensor/Actuator Failures," Proc. IEEE Conf. Dec. and Cont., Tampa, Florida, pp. 1566-1572, Dec. 1989.

10. Stevens, R.D., "Characterization of a Reconfigurable Multiple Model Adaptive Controller Using a STOL F-15 Model," M.S.E.E. thesis, A.F. Inst. of Tech., Wright-Patterson AFB, Ohio, Dec. 1989.

11. Maybeck, P.S., and R.D. Stevens, "Reconfigurable Flight Control Via Multiple Model Adaptive Control Methods," IEEE Transactions on Aerospace and Electronic Systems, Vol AES-27, No. 3, pp. 470-480, May 1991.

12. Stratton, G.L., "Actuator and Sensor Failure Detection Using a Multiple Model Adaptive Technique for the VISTA/F-16", M.S.E.E. thesis, A.F. Inst. of Tech., Wright-Patterson AFB, Ohio, Dec. 1991.

13. Military Specification - Flying Qualities of Piloted Airplanes (MIL-STD-1797A), Government Printing Office, Washington D.C., 30 January 1990.